

```

main(int argc, char *argv[])
{
int i = 0;

/*if (argc != 5)
    {
        printf("Usage Instructions: dialog config_file learn_file\n");
        printf("*** Exiting, goodbye.\n");
        exit(1);
    }*/
if (argc != 2)
    {
        printf("Usage Instructions: d <ini-file>\n");
        printf("*** Exiting, goodbye.\n");
        exit(1);
    }
readini(argv[1]);
formfile = fdoc;
loadStopWords(sdoc);
numPF = loadFormsList(xdoc);
numForms = loadForms(fdoc);
loadData(cfg, lcfg);
Interactive(lcfg);
}

/*****
loaddata : This function will read the configuration files and load the
            information into the relevant global arrays.
*****/
loadData(char *filem, char *file2)
{
char buf[256], word[20];
int i, j, k, l;
int numext;
FILE *fp, *f2;

/***** open configuration file *****/
fp = fopen(filem, "r");

/***** open learn(extended thesaurus) file *****/
f2 = fopen(file2, "r");

```

**PATENT**  
**Docket No.: 4428-4001**

```

prompts = columnTerms = rowTerms = NULL;
scoring = thesaurus = indexList = menuList = NULL;

/* read data in the arrays */
numMenu = loadMenuTree(fp, "[MENUTREE]");
numIndex = readArray(fp, "[PROMPTS]", &prompts, 1, NULL, 0, 0);
numColumn = readArray(fp, "[INDEX]", &columnTerms, 1, &indexList, numIndex, 0);
numOrgRow = readArray(fp, "[THESAURUS]", &rowTerms, 1, &thesaurus, numColumn, 0);
numRow = readArray(f2, "[EXT-THESAURUS]", &rowTerms, 1, &thesaurus, numColumn,
numOrgRow);
numScore = readArray(f2, "[SCORING]", NULL, 0, &scoring, numColumn + 1, 0);

fclose(fp);
fclose(f2);
}

/*****
readArray : Reads the file and fills the rows and columns of the given arrays
*****/
int readArray(FILE *fp, char *head, char ***ch_array, int ccount, int ***int_array, int icount, int
sp)
{
char buf[256];
int i, j, start = 0, wc = 0;
int k, c;
char **tmparray; /*To store the pointers to the words/numbers from the string*/
c = sp;

if (icontains != 0)
    tmparray = (char **)malloc((icontains + 1) * sizeof(char *));

fseek(fp, 0, 0); /* Go to Top */

while (fgets(buf, 255, fp) != NULL) /* read lines till end of file */
    {
    allTrim(buf);
    j = strlen(buf);
    if (buf[j - 1] == '\n') buf[j - 1] = 0;
    if (start)
        {
        if (strlen(buf) == 0) /* if blank line, stop reading */
            break;
        if (icontains == 0) /* i.e. no integer array */

```

```

        addWord(ch_array, buf, ++c);
    else /* read first word string */
        { /* rest are columns of int array */
        wc = readValues(buf, tmparray);
        c++;
        (*int_array) = (int **)realloc(*int_array, c * sizeof(int *));
        (*int_array)[c - 1] = (int *)malloc(ccount * sizeof(int));
        if (ccount != 0)
            addWord(ch_array, tmparray[0], c);
        else
            (*int_array)[c - 1][0] = atoi(tmparray[0]);
        for (k = 1; k < ccount; k++)
            if (k < wc)
                (*int_array)[c - 1][k - ccount] = atoi(tmparray[k]);
            else
                (*int_array)[c - 1][k - ccount] = 0;
        }
    }
    else
        if (!strcmp(head, buf))
            start = 1;
    }
    return c;
}

/*****
loadMenuTree : loads the menutree from file to menuList array
*****/
int loadMenuTree (FILE *fp, char *head)
{
    char buf[256];
    int i, j, start = 0, count = 0;
    fseek(fp, 0, 0);
    while (fgets(buf, 255, fp) != NULL)
        {
        j = strlen(buf);
        if (buf[j - 1] == '\n')
            buf[j - 1] = 0;
        if (start)
            {
            if (strlen(buf) == 0)
                break;
            menuList = (int **)realloc(menuList, (count + 1) * sizeof(int *));

```

```

        menuList[count] = (int *)malloc(3 * sizeof(int));
        sscanf(buf, "%d,%d,%d\n", &menuList[count][0],
&menuList[count][1],&menuList[count][2]);
        count++;
    }
    else
        if (!strcmp(head, buf))
            start = 1;
    }
return count ;
}

```

```

readini(char * filenm)
{
    FILE * fp;
    char buf[80], key[80], value[80], comment[80];
    int cnt;
    if ((fp=fopen(filenm,"r"))==NULL)
    {
        perror(filenm);
        exit(1);
    }
    while (fgets(buf,79,fp)!=NULL)
    {
        sscanf(buf,"%s %s %s",key,value, comment);
        if (!strcmp(key, "sdoc"))
            sdoc=strdup(value);
        if (!strcmp(key, "fdoc"))
            fdoc=strdup(value);
        if (!strcmp(key, "xdoc"))
            xdoc=strdup(value);
        if (!strcmp(key, "cfg"))
            cfg=strdup(value);
        if (!strcmp(key, "lcfg"))
            lcfg=strdup(value);
        if (!strcmp(key, "minprompt"))
            minPromptCount=atoi(value);
        if (!strcmp(key, "timeout"))
            timeout=atoi(value);
    }
}

```

**interactive.c:** This program contains funtions related to user interaction

\*\*\*\*\*

Interactive : function to accept a sentence from the user and then  
generate the response.

-----  
thesaurusFlag = is 1 if there is atleast 1 thesaurus/learned word in query  
updateFlag = is set to 1 if the program needs to learn (i.e. main menu was  
selected during the prompt navigation)

interPrompts = Intersection of prompts  
unionPrompts = Union of prompts  
interUnionPrompts = Intersection of Union  
numInter = number of prompts in InterPrompts  
numInterUnion = num of prompts in Intersection of Union  
numUnion = num of prompts in Union  
numUnknown = num of unknown words

\*\*\*\*\*/

```
#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include "globalvar.h"
#include "arraylib.h"
#include "forms.h"
```

```
#define max(a,b) (a > b)? a: b
#define min(a,b) (a < b)? a: b
#define swap(a,b) (a ^= b, b ^= a, a ^= b)
```

```
extern int numScore, **scoring;
int updateFlag = 0, learnFlag, numQueryList = 0;
FILE *lf, *pf;
char ** uWList=NULL, *queryTerms[50];
int uWNum;
extern int minPromptCount, timeout;
char query[256], **queryList = NULL;
char *affirmWords[] = { "yes", "right", "correct"};
char *negWords[] = { "no", "neither"};
extern char * fdoc;
int otheFlag = 0;
int unknownWords[20], numQuery = 0, numUnknown;
char **uWords; // Added this array to facilitate learning wven if lateral shift
int numUW; // Added this to facilitate learning wven if lateral shift
void sayOther();
```

```

void Interactive(char *flnm)
{
    int InterPrompts[20], unionPrompts[20], t1Prompts[20];
    int interUnionPrompts[20], numInterUnion, t2Prompts[20], numT2;
    int i, j, k, l;
    int start, numUnion, numInter, numT1;
    int n, selectedPrompt, thesaurusFlag = 0;
    char *interlog, *processlog, c;

    numUnknown = numUW = 0;
    for(i=0; i < 20; i++)
        unknownWords[i] = 0;
    uWords=NULL;

    if((interlog = (char *)getenv("TIMEOUT")) != NULL)
        timeout = atoi(interlog);

    if((interlog = (char *)getenv("MINPROMPT")) != NULL)
        minPromptCount = atoi(interlog);

    if((interlog = (char *)getenv("INTERLOG")) == NULL)
        interlog = "test.html";

    if((processlog = (char *)getenv("PROCESSLOG")) == NULL)
        processlog = "process.html";

    signal(SIGALRM, &sayOther);
    lf = fopen(interlog, "w");

    pf = fopen(processlog, "w");
    fprintf(lf, "<HTML>\n<TITLE>%s</TITLE>\n<BODY><FONT SIZE=5>\n", interlog);
    fprintf(pf, "<HTML>\n<TITLE>%s</TITLE>\n<BODY><FONT SIZE=5>\n", processlog);

    system("clear");
    printf("Thank you for calling the New Herald.\n");
    printf("How may we help you?\n\n");
    fprintf(lf, "\nThank you for calling the New Herald.<BR>");
    fprintf(lf, "How may we help you. <P>");
    fgets(query, 255, stdin); /* accept the user input */

    while (1)
        {

```

```

addWord(&queryList, query, ++numQueryList);
numQuery = thesaurusFlag = 0;
if (strlen(query) == 0)
    break;
fprintf(lf, "<I> %s</I> <P>", query);

numQuery = createArray(query, queryTerms);

/*****/
fprintf(pf, "Terms in Query: ");
for (j = 0; j < numQuery; j++)
    fprintf(pf, " %s", queryTerms[j]);
fprintf(pf, "<BR>");
/*****/

/* initialize InterPrompts and unionPrompts array */
for (i = 0; i < 20; i++)
    {
        t2Prompts[i] = t1Prompts[i] = 0;
        InterPrompts[i] = unionPrompts[i] = interUnionPrompts[i] = 0;
    }
start = 1;
numInterUnion = numT2 = numT1 = numInter = numUnion = 0;

/* Scan thru all the words to generate union/intersection of prompts*/
for (i = 0; i < numQuery; i++)
    {
        /* if not in index words check thesaurus */
        if (!inArray(columnTerms, queryTerms[i], numColumn))
            {
                learnFlag = numT1 = numT2 = 0;
                scanThesaurus(queryTerms[i], t1Prompts, t2Prompts, &numT1,
&numT2);

                /* if unknown/learned word save it to array */
                if (learnFlag)
                    {
                        unknownWords[numUnknown] = i, numUnknown++;
                        addWord(&uWords, queryTerms[i], ++numUW);
                        if (numT1 == 0 && numT2 == 0)
                            continue;
                        else
                            thesaurusFlag = 1;
                    }
            }
    }

```

```

else
    thesaurusFlag = 1;
}
else
{
    numT1 = fetchPrompts(queryTerms[i], t1Prompts);
    numT2 = fetchPrompts(queryTerms[i], t2Prompts);
    /*******/
    fprintf(pf, "%s (index) :", queryTerms[i]);
    for (j = 0; j < numT1; j++)
        fprintf(pf, " %d", t1Prompts[j]);
    fprintf(pf, "<BR>");
    fflush(pf);
    /*******/
}

if (start) /* if first word */
{
    numInter = PromptUnion(InterPrompts, t2Prompts, numInter, numT2);
    numUnion = PromptUnion(unionPrompts, t1Prompts, numUnion,
numT1);
    numInterUnion = PromptUnion(interUnionPrompts, t1Prompts,
numInterUnion, numT1);
    start = 0;
}
else
{
    numInter = PromptIntersection(InterPrompts, t2Prompts, numInter,
numT2);
    numUnion = PromptUnion(unionPrompts, t1Prompts, numUnion,
numT1);
    numInterUnion = PromptIntersection(interUnionPrompts, t1Prompts,
numInterUnion, numT1);
}
}
/*******/
fprintf(pf, "Final Intersection Result: ");
for (j = 0; j < numInter; j++)
    fprintf(pf, " %d", InterPrompts[j]);
fprintf(pf, "<BR>");
fprintf(pf, "Final Intersection of Union Result: ");
for (j = 0; j < numInterUnion; j++)

```



```

        fprintf(pf, "%d", interUnionPrompts[j]);
    fprintf(pf, "<BR>");
    fprintf(pf, "Final Union Result: ");
    for (j = 0; j < numUnion; j++)
        fprintf(pf, "%d", unionPrompts[j]);
    fprintf(pf, "<BR>");
    fflush(pf);
/*****

if (numInter < minPromptCount && thesaurusFlag)
    {
        if (numInterUnion < minPromptCount)
            numInter = PromptUnion(InterPrompts, unionPrompts, numInter,
numUnion);
        else
            numInter = PromptUnion(InterPrompts, interUnionPrompts,
numInter, numInterUnion);
    }

    fprintf(pf, "Final Selection : ");
    fflush(pf);
    for (j = 0; j < numInter; j++)
        fprintf(pf, "%d", InterPrompts[j]);
    fprintf(pf, "<BR>");
    fflush(pf);
    numInter = orderPrompts(InterPrompts, numInter);
    numInter = removeChild(InterPrompts, numInter);
    // eliminate prompts > 3
    for (j = 3; j < numInter; j++)
        InterPrompts[j] = 0;
    numInter = min(numInter, 3);
    fprintf(pf, "Selection After Elimination of descendants: ");
    fflush(pf);
    for (j = 0; j < numInter; j++)
        fprintf(pf, "%d", InterPrompts[j]);
    fprintf(pf, "<BR>");
    fflush(pf);
    selectedPrompt = GetPrompt(InterPrompts, numInter);
    if (selectedPrompt == 100)
        continue;

// if (updateFlag)
learnThesaurus(selectedPrompt, unknownWords, numUnknown, flnm);

```

```

        updateFlag = 0;
    for(j = 0; (j < numMenu) && (menuList[j][1] != selectedPrompt); j++);
    if (menuList[j][2] >= 99)
    {
        for (i = 0; i < numPF; i++)
        {
            if (!strcmp(Fprompts[i][1],prompts[selectedPrompt - 1]))
            {
                for(k = 0; k < numForms; k++)
                    if (!strcmp(Fprompts[i][0],formlist[k]->name))
                    {
                        fillForm(formlist[k], queryList, numQueryList);
                        processForm(formlist[k]);
                        break;
                    }
                break;
            }
        }
        if (i == numPF)
        {
            system("clear");
            printf("\nYour query has been understood.\n");
            printf("Please wait to be transferred to the relevant department.\n\n");
            fprintf(If,"<P>Your query has been understood.<LI>");
            fprintf(If,"Please wait to be transferred to the relevant department.<HR>");
            break;
        }
    }
    else
    {
        printf("\n%s\n\n",prompts[menuList[j][2] - 1]);
        fprintf(If, "\n<P>%s<HR>",prompts[menuList[j][2] - 1]);
    }
    // modified for the loop
    printf("Do you have another query?\n\n");
    fgets(query,80,stdin);
    if (!chkNegtn(query))
    {
        for(i = 0; i < numQueryList; i++)
            free(queryList[i]);
        for(i = 0; i < numForms; i++)
            free(formlist[i]);
        free(formlist);
    }

```



```

        learnFlag = 1;
    }
else
    fprintf(pf, "%s (thesaurus):<BR>", word);

for (j = 0; thesaurus[1][j] && j < numRows; j++)
    {
    m = fetchPrompts(columnTerms[thesaurus[1][j] - 1], tp);
    /*****
    fprintf(pf, "<LI>%s (index) :", columnTerms[thesaurus[1][j] - 1]);
    for (k = 0; k < m; k++)
        fprintf(pf, " %d", tp[k]);
    fprintf(pf, "<BR>");
    /*****

    *n1 = PromptUnion(t1Prompts, tp, *n1, m);
    if (sflg)
        {
        *n2 = PromptUnion(t2Prompts, tp, *n2, m);
        sflg = 0;
        }
    else
        {
        *n2 = PromptIntersection(t2Prompts, tp, *n2, m);
        }
    }
    fprintf(pf, "Union Result: ");
    for (k = 0; k < *n1; k++)
        fprintf(pf, " %d", t1Prompts[k]);
    fprintf(pf, "<BR>");
    fprintf(pf, "Intersection Result: ");
    for (k = 0; k < *n2; k++)
        fprintf(pf, " %d", t2Prompts[k]);
    fprintf(pf, "<BR>");
    }

return k;

}
/*****
PromptUnion : does a union of arrays pointed by p1 and p2 and
stores in p1. returns the total elements in result
*****/

```

```

int PromptUnion(int *p1, int *p2, int n1, int n2)
{
    int i, j;
    for (i = 0; i < n2; i++)
        {
            for (j = 0; j < n1; j++)
                if (p1[j] == p2[i])
                    break;
            if (j == n1)
                {
                    p1[j] = p2[i];
                    n1++;
                }
        }
    return n1;
}

```

```

/*****
PromptIntersection : does a intersection of arrays pointed by p1 and p2 and
                    stores in p1. returns the total elements in result
*****/

```

```

int PromptIntersection(int *p1, int *p2, int n1, int n2)

```

```

{
    int i, j;
    for (i = 0; i < n1; i++)
        {
            for (j = 0; j < n2; j++)
                if (p1[i] == p2[j])
                    break;
            if (j == n2) /* not there */
                {
                    for (j = i; j < n1; j++)
                        p1[j] = p1[j + 1];
                    n1--;
                    i--;
                }
        }
    return n1;
}

```

```

/*****
fetchPrompts : Will fetch all the prompts for 'word' into
              Arraylist pointed by t1Prompts;
*****/

```

**PATENT**  
**Docket No.: 4428-4001**

```

int fetchPrompts(char *word, int *t1Prompts)
{
int i, j, k, l;
for (i = 0; i < 20; i++)
    t1Prompts[i] = 0;
if ((i = inArray(columnTerms, word, numColumn)) == 0)
    return 0;
i--;
for (j = 0; (t1Prompts[j] = indexList[i][j]) && (j < numIndex); j++);
return j;
}

/*****
GetPrompt: Returns the final prompt selected by user
*****/
int GetPrompt(int *Parray, int pcnt)
{
int i, j, k, l;
int mmflag = 0, af = 0;
char ans[80];

while (1)
{
system("clear");
printf("\n");
fprintf(If, "<P>");
// Removed the comments to reintroduce last prompt
if (pcnt == 1 && isLeaf(Parray[0]) && numUnknown > 0)
    af = 1;
// -----
if ((pcnt > 1) || (pcnt == 1 && af == 1))
{
// sortPrompts(Parray, pcnt);
orderPrompts(Parray, pcnt);
for (i = 0; i < pcnt; i++)
{
printf("%s\n\n", prompts[Parray[i] - 1]);
fprintf(If, "<LI>%s", prompts[Parray[i] - 1]);
}
if (!mmflag)
{
otheFlag = 0;
alarm(timeout);
}
}
}

```

```

    }
    fgets(ans, 80, stdin); /* accept the user input */
    alarm(0);
    fprintf(1f, "<P><I>%s</I><P>", ans);
    fflush(1f);
    if (otheFlag == 1 && chkAfrm(ans))
        j = 0;
    else
    {
        if (chkNegtn(ans) && otheFlag != 1)
            j = 0;
        else
        {
            addWord(&queryList, ans, ++numQueryList);
            j = chkAns(ans, Parray, pcnt);
        }
        otheFlag = 0;
    }
    if (j == -99)
    {
        updateFlag = 1;
        return 100;
    }
    if (j < 0)
    {
        pcnt = removeZeros(Parray,pcnt);
        continue;
    }
    mmflag = 0;
}
else
    j = pcnt;
if (j == 0)
{
    pcnt = getNodes(j, Parray);
    mmflag = updateFlag = 1;
}
else
{
    if (isLeaf(Parray[j - 1]))
    {
        return Parray[j - 1];
    }
}

```

```

        else
            pcnt = getNodes(Parray[j - 1], Parray);
    }
    af = 1;
}
}

/*****
isLeaf: Returns 1 if 'node' is a leaf in the menutree, else 0
*****/
int isLeaf(int node)
{
    int i;
    for (i = 0; i < numMenu; i++)
        if (menuList[i][1] == node)
            break;
    if (i == numMenu)
        return 0;
    return menuList[i][2];
}

int getNodes(int pnode, int *parray)
{
    int i, j;
    for (i = 0, j = 0; i < numMenu; i++)
        if (menuList[i][0] == pnode)
            {
                parray[j] = menuList[i][1];
                j++;
            }
    parray[j] = 0;
    return j;
}

/*****
learnThesaurus : re-writes the thesaurus with relearned pattern and newly
                learned word.
*****/
learnThesaurus(int pmpt, int unknownWords[], int numUnknown, char *flnm)
{
    int i, j, k, l;
    FILE *fp;
    int *tmpList, tmpCount;

```



```

/* create and initialize a tmp Array */
tmpList = (int *)calloc(numColumn, sizeof(int));
for (i = tmpCount = 0; i < numColumn; i++)
    tmpList[i] = 0;

/* scan thru the query words and gather a list of unique keywords in tmp array*/
tmpCount = getKeyWords(queryTerms, numQuery, tmpList);
/* Locate the row for select prompt. if not create new row */
for (k = 0; k < numScore && scoring[k][0] != pmpt; k++);
if (k >= numScore)
{
    scoring = (int **)realloc(scoring, (k + 1) * sizeof(int *));
    scoring[k] = (int *) malloc((numColumn + 1) * sizeof(int));
    for (j = 0; j <= numColumn; j++)
        scoring[k][j] = 0;
    numScore++;
}

scoring[k][0] = pmpt;
for (j = 0; j < tmpCount; j++)
    scoring[k][tmpList[j]]++;

/*****/
for (i = tmpCount = 0; i < numColumn; i++)
    tmpList[i] = 0;

for (j = i = 0; j < numColumn; j++)
{
    for (k = 0; k < numIndex && indexList[j][k] != 0; k++)
        if (indexList[j][k] == pmpt)
            break;
    if (k < numIndex && indexList[j][k] != 0)
    {
        tmpList[i] = j + 1;
        i++;
    }
}
tmpCount = i;
fp = fopen(flnm, "w");
fprintf(pf, "<BR><B>Learned words</B><BR>");
fprintf(fp, "[%s]\n", "EXT-THESAURUS");

```

```

for (i = numOrgRow; i < numRows; i++)
{
    fprintf(fp, "%s: ", rowTerms[i]);

    if (updateFlag && inArray(uWords, rowTerms[i], numUW))
    {
        fprintf(pf, "%s (relearned)<BR>original: ", queryTerms[unknownWords[j]]);
        for (k = 0; k < numColumn; k++)
        {
            if (thesaurus[i][k] != 0)
                fprintf(pf, " %d", thesaurus[i][k]);
            if (thesaurus[i][k] == 0)
                break;
        }
        fprintf(pf, "<BR>");
        k = PromptUnion(thesaurus[i], tmpList, k, tmpCount);
        fprintf(pf, "new :");
        for (j = 0; j < k; j++)
            fprintf(pf, " %d", thesaurus[i][j]);
        fprintf(pf, "<BR><BR>");
    }

    for (j = 0; j < numColumn; j++)
    {
        if (thesaurus[i][j] == 0)
            break;
        fprintf(fp, "%d,", thesaurus[i][j]);
    }
    fprintf(fp, "\n");
}
for (i = 0; updateFlag && i < numUW; i++)
{
    if (inArray(rowTerms, uWords[i], numRows))
        continue;

    fprintf(fp, "%s: ", uWords[i]);
    fprintf(pf, "%s(new-learned) :", uWords[i]);
    addWord(&rowTerms, uWords[i], ++numRow);
    thesaurus = (int **)realloc(thesaurus, numRows * sizeof(int *));
    thesaurus[numRow - 1] = (int *)malloc(numColumn * sizeof(int));
    for (j = 0; j < numColumn; j++) thesaurus[numRow - 1][j] = 0;
    for (j = 0; j < tmpCount; j++)
    {

```

PATENT  
Docket No.: 4428-4001

```

        thesaurus[numRow - 1][j] = tmpList[j];
        fprintf(fp, "%d,", tmpList[j]);
        fprintf(pf, " %d", tmpList[j]);
    }
    fprintf(fp, "\n");
    fprintf(pf, "<BR><BR>");
}

fprintf(fp, "\n");

/* write the scoring in the file */
fprintf(fp, "[%s]\n", "SCORING");
for (i = 0; i < numScore; i++)
    {
        fprintf(fp, "%d,", scoring[i][0]);
        for(j = 1; j <= numColumn; j++)
            fprintf(fp, "%d,", scoring[i][j]);
        fprintf(fp, "\n");
    }
fprintf(fp, "\n");
fclose(fp);
}

/*****
removeChild: removes descendents of all the elements from the list
*****/
int removeChild(int *array, int tot)
{
    int i, j, k, cnt = 0;
    int *tmparray, rn = 99;

    tmparray = (int *)calloc(numIndex, sizeof(int));

    /* Remove any prompts that are responses rather than choices */
    for (i = 0; i < tot; i++)
        {
            for(j = 0; j < numMenu; j++)
                if (menuList[j][1] == array[i] && menuList[j][2] == 100)
                    array[i] = 0;
        }

    /* Remove any prompts that are root node and have a child which is not a leaf rather than
    choices */

```

```

for (i = 0; i < tot; i++)
{
    if (array[i] == 0) /* already removed so go to next */
        continue;

    /* if (isLeaf(array[i]))
        continue; */

    rn = array[i];
    while(1)
    {
        for(j = 0; j < numMenu; j++)
            if (menuList[j][1] == rn)
                break;
        if (menuList[j][0] == 0)
            break;
        rn = menuList[j][0];
    }
    if (rn != array[i])
    {
        for (j = 0; j < tot; j++)
        {
            if (array[j] == rn)
                array[j] = 0;
        }
    }
}

for (i = 0; i < tot; i++)
{
    if (array[i] == 0) /* already removed so go to next */
        continue;
    for (j = 0; j < numIndex; j++) /* initialize tmparray */
        tmparray[j] = 0;
    cnt = getChildren(array[i], tmparray); /* get children & grand-children of i */
    for (j = 0; j < tot; j++) /* scan thru the array to check for child */
        if (j != i) /* ignore self from checking */
            for (k = 0; k < cnt; k++)
                if (array[j] == tmparray[k])
                {
                    array[j] = 0; /* if j is child of i, make it 0 */
                    break;
                }
}

```

```

    }
}
/* Shift All non-zeroes upwards */
for (i = 0; i < tot; i++)
{
    if (array[i] == 0)
    {
        for (j = i + 1; j < tot; j++)
            if (array[j] != 0)
                break;
        if (j < tot)
        {
            array[i] = array[j];
            array[j] = 0;
        }
    }
}
/* count no of elements */
for (j = 0; j < tot; j++)
{
    if (array[j] == 0)
        break;
}
return j;
}

/*****
getChildren: fetches all the descendents of pmpt into array
*****/
int getChildren(int pmpt, int *array)
{
    int i, j, k, l;
    int t, t1, t2;
    int *tmparray1, *tmparray2;

    if (isLeaf(pmpt)) /* if node is leaf no children so return 0 */
        return 0;

    tmparray1 = (int *)calloc(numIndex, sizeof(int)); /* child of child in every loop */
    tmparray2 = (int *)calloc(numIndex, sizeof(int)); /* union of all scanned children */
    t = t1 = t2 = 0;
    for (i = 0; i < numMenu; i++)
    {

```

**PATENT**  
**Docket No.: 4428-4001**

```

    if (menuList[i][0] == pmpt && menuList[i][2] != 100)
        {
            array[t] = menuList[i][1];
            t1 = getChildren(array[t], tmparray1);
            t2 = PromptUnion(tmparray2, tmparray1, t2, t1);
            t++;
        }
    }
t = PromptUnion(array, tmparray2, t, t2);
return t;
}

int chkAns (char * ans, int * Parray, int pcnt)
{
char locquery[256];
int i, j, tmp1cnt = 0, tmp2cnt = 0;
char *resWords[50], start = 'Y';
int numWords, tmpArray1[20], tmpArray2[20];
int uwFlag = 0, rowOrColWord = 0;

strcpy(locquery,ans);
numWords = breakStr(ans, resWords);
if ( strcmp(resWords[0],"other") == 0 && strcmp(resWords[1],"options") == 0)
    {
        return 0;
    }
numWords = processArray(resWords, numWords, 1);
for (i = 0; i < 20; i++)
    tmpArray1[i] = tmpArray2[i] = 0;
fprintf(pf,"<i>Initialized Temp Array\n"); fflush(pf);

for (i = 0; i < numWords; i++)
    {
        if (!inArray(columnTerms, resWords[i], numColumn))
            {
                if (!inArray(rowTerms,resWords[i],numColumn))
                    {
                        if (!inArray(uWList, resWords[i], uWNum))
                            {
                                addWord(&uWList, resWords[i], ++uWNum);
                                fflush(lf);
                            }
                    }
            }
    }

```

PATENT  
Docket No.: 4428-4001

```

        else
            {
                fprintf(pf,"<li>Unknown Word: %s\n",resWords[i]);
                fflush(pf);
                uwFlag = 1; /* unKnown word encountered twice */
            }
        }
    else
        rowOrColWord++;
    continue;
}
else
    rowOrColWord++;
tmp1cnt = fetchPrompts(resWords[i], tmpArray1);
if (start == 'Y')
    {
        tmp2cnt = PromptUnion(tmpArray2, tmpArray1, tmp2cnt, tmp1cnt);
        start='N';
    }
else
    tmp2cnt = PromptIntersection(tmpArray2, tmpArray1, tmp2cnt, tmp1cnt);

tmp2cnt = PromptIntersection(tmpArray2, Parray, tmp2cnt, pcnt);
}
if (tmp2cnt != 1)
    {
not selected
        if (tmp2cnt == 0 && pcnt == 1 && numWords == 1) // i.e. only one prompt &
        {
            strcpy(ans, locquery);
            if (chkAfrm(ans))
                return 1;
        }
        if (tmp2cnt > 1) // i.e. multiple prompt selection then do score
        {
            strcpy(ans, locquery);
            return checkscore(ans, Parray, pcnt);
        }
    }

if (uwFlag)
    if (AskforOp())
        return -99;

```

```

        else
            return -1;
    else
        if (rowOrColWord)
            {
                strcpy(query,locquery);
                return -99;
            }
    }
    for (i = 0; Parray[i]; i++)
        if (Parray[i] == tmpArray2[0])
            return i + 1;
}

int AskforOp()
{
    int i, j;
    char *resWords[50];
    int numWords ;

    system("clear");
    printf("Your request was not understood.\n");
    printf("Would you prefer to speak to an operator or try again with a new request?\n");
    fprintf(lf, "<P>Your request was not understood.<LI>");
    fprintf(lf, "Would you prefer to speak to an operator or try again with a new request?\n");
    fflush(lf);
    fgets(query, 255, stdin); /* accept the user input */
    addWord(&queryList, query, ++numQueryList);
    fprintf(lf, "<P><I> %s</I>", query);
    numWords = breakStr(query, resWords);
    if ( inArray(resWords,"operator",numWords))
        {
            printf("\n\nPlease wait for the operator ... \n");
            fprintf(lf,"<P>Please wait for the operator ...");
            fflush(lf);
            exit(0);
        }
    if( inArray(resWords,"try",numWords) && inArray(resWords,"again",numWords))
        {
            system("clear");
            printf("Please tell us your new request\n");
            fprintf(lf, "<P>Please tell us your new request\n");
            fflush(lf);

```



**PATENT**  
**Docket No.: 4428-4001**

```

    fgets(query, 255, stdin); /* accept the user input */
    addWord(&queryList, query, ++numQueryList);
}
return 1;
}

void sayOther()
{
printf("\nWould you like to hear other options?\n\n");
fprintf(If, "<LI>Would you like to hear other options?<P>");
otheFlag = 1;
}

int checkscore(char *ans, int *Parray, int pcnt)
{
    char * resWords[50], *pmptWords[50];
    int i, j, *score, *score1;
    int numWords, numpWords, maxscore;

    score = (int *)malloc(pcnt * sizeof(int));
    score1 = (int *)malloc(pcnt * sizeof(int));
    for (i = 0; i < pcnt; i++)
        score[i] = score1[i] = 0;

    numWords = breakStr(ans, resWords);
    for (i = 0; i < pcnt; i++)
    {
        numpWords = breakStr(prompts[Parray[i]-1], pmptWords);
        for (j = 0; j < numWords; j++)
            if (inArray(pmptWords, resWords[j], numpWords))
                score[i]++;
    }
    for (i = maxscore = 0; i < pcnt; i++)
        maxscore = (maxscore < score[i])?score[i]:maxscore;
    for (i = j = 0; i < pcnt; i++)
        j += (score[i] == maxscore)?1:0;

    if (j == 1) /* single prompt selection */
    {
        for (i = 0; i < pcnt; i++)
            if (score[i] == maxscore)
                return i + 1;
    }
}

```

```

else
{
    for (i = 0; i < pcnt; i++)
    {
        numpWords = breakStr(prompts[Parray[i] - 1], pmptWords);
        score1[i] = getscore1(resWords, numWords, pmptWords, numpWords);
    }
    maxscore = 0;
    for (i = 0; i < pcnt; i++)
        maxscore = (score1[i] > maxscore)?score1[i]:maxscore;
    for (i = j = 0; i < pcnt; i++)
        j += (score1[i] == maxscore)?1:0;
    if (j == 1) /* single prompt selection */
    {
        for (i = 0; i < pcnt; i++)
            if (score1[i] == maxscore)
                return i + 1;
    }
    else
    {
        for (i = 0; i < pcnt; i++)
            if (score[i] != maxscore)
                Parray[i] = 0;
        return -1;
    }
}

}

int chkAfrm( char * str)
{
    int i,j, numWords;
    char * resWords[50];

    numWords = breakStr(str, resWords);
    for (i = 0; i < numWords; i++)
        if (inArray(affrmWords, resWords[i], 3))
            return 1;

    return 0;
}

int chkNegtn( char * str)
{

```

```

int i,j, numWords;
char * resWords[50];

numWords = breakStr(str, resWords);
for (i = 0; i < numWords; i++)
    if (inArray(negWords, resWords[i], 2))
        return 1;
return 0;
}

int getscore1(char **Word1, int num1, char **Word2, int num2)
{
    int i, j, scr = 0;
    int lsmatch = 0;

    for(i = 0; i < num1; i++)
    {
        for(j = lsmatch; j < num2; j++)
            if (!strcmp(Word1[i], Word2[j]))
            {
                scr++;
                break;
            }
        if (j < num2)
            lsmatch = j + 1;
    }

    return scr;
}

int orderPrompts(int *InterPrompts, int numInter)
{
    int i, j, k, l;
    int *tmpArray[2]; /* 0 - score ; 1 - level; 3 - menu order */
    int *tmpList, tmpCount;

    tmpArray[0] = (int *)malloc(numInter * sizeof(int));
    tmpArray[1] = (int *)malloc(numInter * sizeof(int));

    tmpList = (int *)malloc(numColumn * sizeof(int));
    for (i = 0; i < numColumn; i++)
        tmpList[i] = 0;
    // get the list of keywords from queryTerms

```

```

tmpCount = getKeyWords(queryTerms, numQuery, tmpList);

for (i = 0; i < numInter; i++)
{
    /* get the maxscore for the prompt */
    tmpArray[0][i] = 0;
    for (j = 0; (j < numScore) && (scoring[j][0] != InterPrompts[i]); j++);
    /* if any previous scoring present */
    if ((j < numScore) && (scoring[j][0] == InterPrompts[i]))
    {
        // get the max score
        for (k = 0; k < tmpCount; k++)
            tmpArray[0][i] = max(tmpArray[0][i], scoring[j][tmpList[k]]);
    }
    tmpArray[1][i] = getLevel(InterPrompts[i]);
}

// sort the array in order of score, level and menu-order
for (i = 0; i < (numInter - 1); i++)
{
    for (j = i + 1; j < numInter; j++)
        if (!gThan(tmpArray[0][i], tmpArray[1][i], InterPrompts[i],
tmpArray[0][j], tmpArray[1][j], InterPrompts[j]))
        {
            swap(tmpArray[0][i], tmpArray[0][j]);
            swap(tmpArray[1][i], tmpArray[1][j]);
            swap(InterPrompts[i], InterPrompts[j]);
        }
}

return numInter;
}

int getKeyWords(char **queryTerms, int numQuery, int *tmpList)
{
    int i, j, k, l;
    int count = 0;

    for (j = 0; j < numQuery; j++)
    {
        /* Check if the word is keyword */
        if ((k = inArray(columnTerms, queryTerms[j], numColumn)) != 0)
        {

```

```

        /* add in temp list only if not present */
        for (i = 0; i < count && tmpList[i] != k ; i++);
        if (i >= count)
            tmpList[count++] = k;
        continue;
    }

    /* check if the word is Thesaurus/Learned Word */
    if ((k = inArray(rowTerms, queryTerms[j], numRows)) != 0)
    {
        /* pick-up all keywords for that word */
        for (i = 0; thesaurus[k - 1][i] != 0; i++)
        {
            for (l = 0; l < count && tmpList[l] != thesaurus[k - 1][i] ; l++);
            if (l >= count)
                tmpList[count++] = thesaurus[k - 1][i];
        }
    }
}

return count;
}

int getLevel(int pmpt)
{
    int i, k, l;

    for (i = 0; i < numMenu && menuList[i][1] != pmpt; i++);
    k = menuList[i][0];
    for (l = 0; k > 0; l++)
    {
        for (i = 0; i < numMenu && menuList[i][1] != k; i++);
        k = menuList[i][0];
    }
    return l;
}

int gThan(int a, int b, int c, int p, int q, int r)
{
    if (a > p) return 1; // Desc order here
    if (a < p) return 0; // Desc order here
    if (b > q) return 0; // Asc order here
}

```

```

    if (b < q) return 1; // Asc order here
    if (c > r) return 0; // Asc order here
    return 1; // Asc order here
}

```

**formlib.c:** This program contains functions for forms processing

---

```

#include <stdio.h>
#include <string.h>
#include "arraylib.h"

struct input {
char *Type;
char *APrompt;
char *RPrompt;
char *Name;
char *Value;
char **Choice;
int numChoice;
};

struct form {
char * name;
struct input **fields;
int numFields;
};

char * split(char * , char );

int loadForm(FILE *f, struct form *frm, char *name)
{
int j, start=0;
char buf[512];
char fname[20];

sprintf(fname, "[%s]", name);
fseek(f, SEEK_SET, 0);
while(fgets(buf, 512, f) != NULL) {
    j = strlen(buf);
    if (buf[j - 1] == '\n') buf[j - 1] = 0;
    if (start)
        {

```

**PATENT**  
**Docket No.: 4428-4001**

```

        if (strlen(buf) == 0) /* if blank line, stop reading */
            break;
        frm->numFields++;
        frm->fields = (struct input **)realloc(frm->fields, (frm->numFields) *
sizeof(struct input *));
        frm->fields[frm->numFields-1] = (struct input *)malloc(sizeof(struct input));
        loadInput(frm->fields[frm->numFields-1], buf);
    }
    else
        if (!strcmp(fname, buf)) {
            start = 1;
            frm->name = strdup(name);
            frm->numFields=0;
            frm->fields=NULL;
        }
    }
    return start;
}

```

```

loadInput(struct input *inp, char * str)
{
    char ***list, *tmpstr1, *tmpstr2;
    int i, j, len;

```

```

    inp->Type = inp->APrompt = inp->RPrompt = inp->Name = inp->Value = NULL;
    list = (char ***)malloc(2 * sizeof(char **));
    list[0] = (char **)malloc(2 * sizeof(char *));
    list[1] = (char **)malloc(2 * sizeof(char *));
    list[0][0] = str;
    for(i=0;(list[i+1][0] = split(list[i][0],':'))!=NULL;i++)
    {
        list[i][1] = split(list[i][0],'=');
        list = (char ***)realloc(list,(i+3)*sizeof(char**));
        list[i+2] = (char **)malloc(2 * sizeof(char *));
    }
    list[i][1] = split(list[i][0],'=');
    len = i + 1;
    for(i=0; i <len;i++)
    {
        if (!strcmp("Type",list[i][0]))
            mystrcp(&inp->Type,list[i][1]);
        if (!strcmp("APrompt",list[i][0]))
            mystrcp(&inp->APrompt,list[i][1]);

```

```

if (!strcmp("RPrompt",list[i][0]))
    mystrcp(&inp->RPrompt,list[i][1]);
if (!strcmp("Name",list[i][0]))
    mystrcp(&inp->Name,list[i][1]);
if (!strcmp("Value",list[i][0]))
    mystrcp(&inp->Value,list[i][1]);
if (!strcmp("Choice",list[i][0]))
    {
        mystrcp(&tmpstr1, list[i][1]);
        tmpstr2 = tmpstr1;
        inp->Choice = NULL;
        inp->numChoice=0;
        for(j=0;tmpstr1[j];j++)
            {
                if (tmpstr1[j]=='')
                    {
                        tmpstr1[j]=0;
                        inp->Choice = (char **)realloc(inp->Choice,(inp-
>numChoice+1)*sizeof(char *));
                        inp->Choice[inp->numChoice++] = strdup(tmpstr2);
                        allTrim(inp->Choice[inp->numChoice-1]);
                        tmpstr2=tmpstr1+j+1;
                    }
            }
        inp->Choice = (char **)realloc(inp->Choice,(inp-
>numChoice+1)*sizeof(char *));
        inp->Choice[inp->numChoice++] = strdup(tmpstr2);
        allTrim(inp->Choice[inp->numChoice-1]);
    }
}

mystrcp(char **str1, char *str2)
{
int len, i, j;

len = strlen(str2);
if(str2[0]=="" && str2[len-1]=="" // i.e. quoted string;
    for (i = str2[--len] = 0; (str2[i] = str2[i + 1]); i++);
*str1 = (strlen(str2)==0)?NULL:strdup(str2);
}

char * split(char * str, char dlm)

```



```

{
int i;
for (i = 0; str[i]; i++)
    if (str[i] == dlm)
        {
            str[i] = 0;
            return str + i + 1;
        }
return NULL;
}

```

```

acceptForm(struct form *frm)
{
int i;
char ans[256];
struct input cnfm;

```

```

cnfm.Type = "MChoice";
cnfm.APrompt = strdup("Is this information correct?");
cnfm.numChoice = 4;
cnfm.Choice = (char **)malloc(2 * sizeof(char *));
cnfm.Choice[0] = strdup("no");
cnfm.Choice[1] = strdup("yes");
cnfm.Choice[2] = strdup("right");
cnfm.Choice[3] = strdup("correct");
cnfm.Value = NULL;

```

```

system("clear");
printf("\n");
for(i = 0; i < frm->numFields; i++)
    {
        if (!strcmp(frm->fields[i]->Type,"Say"))
            sayText(frm->fields[i]);
        if (frm->fields[i]->Value != NULL)
            continue;
        if (!strcmp(frm->fields[i]->Type,"AcceptResponse"))
            getText(frm->fields[i]);
        if (!strcmp(frm->fields[i]->Type,"MChoice"))
            getChoice(frm->fields[i]);
    }
while (1)
    {
        system("clear");

```

```

printf("\n");
for(i = 0; i<frm->numFields; i++)
{
    if (!strcmp(frm->fields[i]->Type,"AcceptResponse"))
        sayText(frm->fields[i]);
    if (!strcmp(frm->fields[i]->Type,"MChoice"))
        sayText(frm->fields[i]);
}
printf("\n");
getChoice(&cnfm);
if (strcmp(cnfm.Value,"no"))
    return 1;
system("clear");
printf("\n");
for(i = 0; i<frm->numFields; i++)
{
    if (!strcmp(frm->fields[i]->Type,"AcceptResponse"))
        getText(frm->fields[i]);
    if (!strcmp(frm->fields[i]->Type,"MChoice"))
        getChoice(frm->fields[i]);
}
}
}

```

```

getText(struct input * inp)
{
    char buf[256];
    printf("\n%s\n\n",inp->APrompt);
    fgets(buf,255,stdin);
    allTrim(buf);
    inp->Value = strdup(buf);
}

```

```

sayText(struct input * inp)
{
    if (inp->RPrompt != NULL)
        printf("%s",inp->RPrompt);
    if (inp->Value != NULL)
        printf("%s",inp->Value);
    if (inp->RPrompt != NULL || inp->Value != NULL)
        printf("\n");
}

```

**PATENT**  
**Docket No.: 4428-4001**

```

fillForm(struct form * frm, char ** Array, int arrCount)
{
int i, j, wrdCount = 0, tmpCount = 0;
char **wordList = NULL;
char *tmparray[50];
for(i = 0; i < arrCount; i++)
    {
        tmpCount = breakStr(Array[i], tmparray);
        wrdCount = mergeArray(&wordList,tmparray, wrdCount, tmpCount);
    }
wrdCount = processArray(wordList, wrdCount, 1);
for(i = 0; i < frm->numFields; i++)
    if(!strcmp(frm->fields[i]->Type,"MChoice"))
        selectValue(frm->fields[i], wordList, wrdCount);
}

```

```

int selectValue(struct input * inp, char **array, int arrCount)
{
int i, j, *score;
char *tmparray[20] ;
int max, maxcount, tmpCount;

score = (int *) malloc(inp->numChoice * sizeof(int));
for (i = 0; i < inp->numChoice; i++)
    {
        score[i] = 0;
        tmpCount = breakStr(inp->Choice[i], tmparray);
        if (tmpCount > 1) // Basically to avoid filtering of 'yes', 'no' etc
            filterStopWords(tmparray, tmpCount);
        tmpCount = processArray(tmparray, tmpCount, 0);
        for(j = 0; j < tmpCount; j++)
            if (inArray(array, tmparray[j], arrCount))
                score[i]++;
    }
for(i = max = 0; i < inp->numChoice; i++)
    if (score[i] > max) max = score[i];
for(i = maxcount = 0; i < inp->numChoice; i++)
    if (score[i] == max) maxcount++;
if (maxcount != 1)
    return 0;
for(i = 0; i < inp->numChoice; i++)
    if (score[i] == max)
        {

```

```

        inp->Value = strdup(inp->Choice[i]);
        break;
    }
return 1;
}

processForm (struct form *frm)
{
    int i, j;
    char *formType = NULL, *formAction = NULL;

    for(i = 0; i < frm->numFields; i++)
    {
        if (frm->fields[i]->Value == NULL)
            continue;
        if (!strcmp(frm->fields[i]->Type, "FormType"))
            formType = strdup(frm->fields[i]->Value);
        if (!strcmp(frm->fields[i]->Type, "FormAction"))
            formAction = strdup(frm->fields[i]->Value);
    }

    // If not defined the form type use 'AcceptForm' as default.
    if (formType == NULL)
        formType = strdup("AcceptForm");

    if (!strcmp(formType, "AcceptForm"))
        j = acceptForm(frm);

    if (!strcmp(formType, "ResponseForm"))
        j = responseForm(frm);

    if (j != 0 && formAction != NULL)
        performAction(frm, formAction);
}

responseForm(struct form *frm)
{
    int i;

    system("clear");
    printf("\n");
    for (i = 0; i < frm->numFields; i++)
    {

```

```

        if (strcmp("Say",frm->fields[i]->Type))
            continue;
        sayText(frm->fields[i]);
    }
printf("\n");
}

getChoice(struct input * inp)
{
char buf[256], *tmparray[50];
int tmpCount;

while (1)
    {
    printf("\n%s\n", inp->APrompt);
    fgets(buf,255,stdin);
    tmpCount = breakStr(buf, tmparray);
    if (tmpCount > 1) // Basically to avoid filtering of 'yes', 'no' etc
        filterStopWords(tmparray, tmpCount);
    tmpCount = processArray(tmparray, tmpCount, 0);
    if (selectValue(inp, tmparray, tmpCount))
        return;
    }
}

performAction(struct form *frm, char *action)
{
struct form f;
char * cmd = NULL;
char buf [256];
int i, j, len1, len2;
FILE *pd;

sprintf(buf, "%s <<EOD\n", action);
cmd = strdup(buf);
for (i = 0; i < frm->numFields; i++)
    {
    if (frm->fields[i]->Name == NULL)
        continue;
    sprintf(buf,"%s=%c%s%c\n",frm->fields[i]->Name,"",frm->fields[i]->Value,"");
    len1 = strlen(buf);
    len2 = strlen(cmd);
    cmd = (char *) realloc(cmd, (len1 + len2 + 1) * sizeof(char));

```

```

        strcat(cmd, buf);
    }
    sprintf(buf, "EOD\n");
    len1 = strlen(buf);
    len2 = strlen(cmd);
    cmd = (char *) realloc(cmd, (len1 + len2 + 1) * sizeof(char));
    strcat(cmd, buf);
    if ((pd = popen(cmd, "r")) == NULL)
    {
        fprintf(stderr, "Error in command execution\n");
        exit(1);
    }
    f.name = NULL;
    f.numFields = 0;
    f.fields = NULL;
    while ((fgets(buf, 255, pd) != NULL))
    {
        j = strlen(buf);
        if (buf[j - 1] == '\n') buf[j - 1] = 0;
        if (strlen(buf) == 0) /* if blank line, stop reading */
            continue;
        f.numFields++;
        f.fields = (struct input **)realloc(f.fields, (f.numFields) * sizeof(struct input *));
        f.fields[f.numFields-1] = (struct input *)malloc(sizeof(struct input));
        loadInput(f.fields[f.numFields-1], buf);
    }
    pclose(pd);
    processForm(&f);
}

```

## HEADER FILES (C)

**globalvar.h:** Header file for global variables

---

```

extern FILE *webDoc, *phoneDoc;
extern int numColumn, numRows, numIndex, numMenu;
extern int startPoint, eofFlag, topValues;
extern char **rowTerms, **columnTerms, **prompts, **stopWords;
extern double **matrix, **cosine;
float phoneThreshold, webThreshold;
extern int **indexList, **menuList, **thesaurus;

```

```
extern int numStopWord, numOrgRow;
void stemArray(char **list, int arrayLen);
extern int numForms, numPF;
int stemWord(char *);
extern char ***Fprompts;
extern int numForms, numPF;
extern struct form **formlist;
```

**process.h:** Header file declaring functions in process.c

```

/*****\
* Process.h:                                     *
\*****/

int processFile(char *filename, char ***cArray, float threshold);
void loadStopWords( char * filename) ;
// int allTrim( char *str);
void fillIndex();
void updateThesaurus( char *str, int pmpt);
void createMatrix(char * filename);
// int readPara(FILE *fp );
// int wordsInPara (FILE *fp);
void calcCosine();
int eraseZeroes();
void createThesaurus();
// void floatSort(int *colnum, float *tmpcos, int numRows);
void saveData(char *filenm);
```

**arraylib.h:** Header file declaring functions in arraylib.c

```

/*****\
* ArrayLib.h                                     *
\*****/

int fetchWord(FILE *f, char * wrd);
int inArray(char **array, char *word, int length);
int removeNulls(char **strarray, int numWords);
int mergeArray(char ***Array1, char **Array2, int numArray1, int numArray2);
int readValues(char *str, char **array);
void sortArray(char *allwords[], int numwords);
int loadPrompts(char *filename);
void loadStopWords(char *);
```

```

FILE * fopen( char *filename, char *mode);
void addWord(char *** cArray, char * word, int c);
int breakStr(char * str, char **strarray);
void filterStopWords(char ** strarray, int numWords);
void filterDuplicates(char ** strarray, int numWords);
int loadFormsList( char *filename);
int loadForms(char * filename);
int allTrim( char *str);
int createArray(char *, char **);
int processArray( char **, int, int);

```

**forms.h:** Header file declaring functions in formlib.c

---

```

#include <stdio.h>

```

```

extern struct input {
char *Type;
char *APrompt;
char *RPrompt;
char *Name;
char *Value;
char **Choice;
int numChoice;
} a;

```

```

extern struct form {
char * name;
struct input **fields;
int numFields;
} b;

```

```

char * split(char *, char );
int loadForm(FILE *, struct form *, char *);

```

```

void loadInput(struct input *, char * );
void mystrcp(char **, char *);
void dumpInput(FILE *, struct input *);
void dumpForm(FILE *, struct form *);
void acceptForm(struct form *);
void getText(struct input * );
void fillForm(struct form * frm, char ** Array, int arrCount);

```



**MAKE FILE**

**makefile:** Makefile for compiling the source code.

---

```

all: t d demorun
t: main.o process.o arraylib.o stemlib.o formlib.o
    cc -g main.o process.o arraylib.o stemlib.o formlib.o -o t -lm
d: dialog.o interactive.o arraylib.o stemlib.o formlib.o
    cc -g dialog.o interactive.o arraylib.o stemlib.o formlib.o -o d
demorun: demorun.c
    cc demorun.c -o demorun
main.o: main.c process.h arraylib.h forms.h
    cc -c -g main.c
process.o: process.c globalvar.h forms.h
    cc -c -g process.c
arraylib.o: arraylib.c globalvar.h
    cc -c -g arraylib.c
dialog.o: dialog.c arraylib.h
    cc -c -g dialog.c
interactive.o: interactive.c globalvar.h
    cc -c -g interactive.c
stemlib.o: stemlib.c
    cc -c -g stemlib.c
formlib.o: formlib.c
    cc -c -g formlib.c
clean:
    rm -f *.o t d core demorun
bkup: clean
    tar cvzf ../stem`date "+%d%m"` .tgz .

```

**PARAMETER FILES**

**t.ini:** This file contains parameters required for program 't'

---

```

pdoc p # phonedoc
wdoc w # webdoc
sdoc s # stopwords
fdoc f # forms
xdoc x # link of forms & prompts
cfg z.cfg # config file
pt 0.02 # phoneThreshold
wt 0.0006 # webThreshold
tv 5 # topValues for cosine

```

**PATENT**  
**Docket No.: 4428-4001**

**d.ini:** This file contains parameters required for program 'd'

---

cfg t.cfg # config file  
lcfg l.cfg # learn file  
sdoc s # stopwords  
fdoc f # forms  
xdoc x # x  
minprompt 2 # minimum no of prompts  
timeout 30 # timeout secs for other options

## DATA FILES

**p:** Document 'p'

---

Are you calling about subscriptions?

    Would you like to order a subscription?

    Would you like to pay your subscription fees?

    Would you like to give a gift subscription?

    Would you like to change your address or change any other information?

    Do you have any billing enquiries or concerns?

        Would you like information about your account balance or your payments?

        Would you like to speak to a customer care representative?

    Would you like to temporarily suspend your delivery?

Is there a problem with your paper or delivery?

    Did you miss today's paper?

    Did you miss yesterday's newspaper and would you like credit for yesterday?

    Did you receive a wet paper?

Would you like information about the New Herald website?

    Would you like to obtain your New Herald website password?

    The website address is [www.newherald.com](http://www.newherald.com). Would you like any other information about the website?

Are you calling about advertisements?

    Would you like to advertise in the New Herald?

        Is it a classified ad?

        Is it a full-page, half-page, or quarter-page ad?

    Would you like to place an ad?

        Is it a classified ad?

        Is it a full-page, half-page, or quarter-page ad?

Are you calling about something else?

    Would you like to write to the New Herald?

        Would you like to submit an article to the op-ed page?

            Please email your article to [oped@newherald.com](mailto:oped@newherald.com).

        Would you like to send a letter to the editor?

**PATENT**  
**Docket No.: 4428-4001**

Please email your letter to [letters@newherald.com](mailto:letters@newherald.com).

Would you like to work for the New Herald?

Would you like to write for the New Herald?

Would you like to work for the editorial division or for the administrative  
division?

w: Document 'w'

---

Now, it's easier than ever to manage your Herald. Welcome to The New Herald Subscription & Customer Care Web site. You expect all the news that's fit to print in each issue of The New Herald. And you can expect responsive, round-the-clock on-line customer care that allows you to review and update your delivery and billing information, stop delivery of your newspaper when you're away, discover special promotions and notify us of any questions or comments you have. And if you're not a subscriber, browse our Web site and consider subscribing to home delivery. Please enjoy your visit.

With convenient home delivery, you will be sure to receive all the wit, the wisdom, the news, the views offered in every issue of The New Herald. And, through this special offer, you will get 50% off the regular rate for the first eight weeks. To subscribe enter your ZIP code below and submit.

ZIP Code:

Expect the World Around the Clock

We are pleased to offer our subscribers instant, 24-hour on-line customer care to meet your service needs. Now, it's easier than ever to order home delivery, review your bill, or change your service -- and to find out about special customer benefits and promotions.

Take Our Survey

Help us provide you with the highest quality customer service. This short survey asks for vital information about you and your service needs. The New Herald may perform statistical analysis of reader interests to identify ways to improve our services and products to better meet the needs of our subscribers. Personal information about you as an individual subscriber will not be provided to any third party. Our privacy policy is posted online to disclose our guidelines for the use of customer information.

You can handle most of your subscription service requests online, including:

- \* Suspending your delivery while you're away

**PATENT**  
**Docket No.: 4428-4001**

- \* Reporting missed deliveries
- \* Checking the status of your account
- \* Checking your billing history
- \* Changing your delivery or billing address
- \* Changing your method of payment

To subscribe

Customer care:

Account Summary

Update Account

Activity History

Billing History

Paper not received

Suspend delivery

Complaints

Order Home Delivery at 50% Off (US Customers Only)

With convenient home delivery, you can start each day with all the news, the views, the wit and the wisdom you expect from The New Herald. And, through this special offer, you can save 50% on the first eight weeks when you order today. It's a smart, easy way to keep up with The Herald.

This offer expires December 31, 2001 and is valid in areas served by The New Herald' delivery service. Subscribers who have had Herald home delivery within the past 90 days are not eligible for this introductory offer. To subscribe enter your ZIP code below and submit.

#### GIFT SUBSCRIPTION OFFER

There's no present like The Herald. And, when you order a gift subscription of 12-week home delivery of The New Herald, you'll save 50% on the regular rate. Hurry. This offer expires

**PATENT**  
**Docket No.: 4428-4001**

December 31, 2001. To order, enter the ZIP code of the gift recipient below and submit.

Order Home Delivery at 50% Off (US Customers Only)

With convenient home delivery, you can start each day with all the news, the views, the wit and the wisdom you expect from The New Herald. And, through this special offer, you can save 50% on the first eight weeks when you order today. It's a smart, easy way to keep up with The Herald.

This offer expires December 31, 2001 and is valid in areas served by The New Herald' delivery service. Subscribers who have had Herald home delivery within the past 90 days are not eligible for this introductory offer. To subscribe enter your ZIP code below and submit.

ZIP Code:

#### GIFT SUBSCRIPTION OFFER

There's no present like The Herald. And, when you order a gift subscription of 12-week home delivery of The New Herald, you'll save 50% on the regular rate. Hurry. This offer expires December 31, 2001. To order, enter the ZIP code of the gift recipient below and submit.

ZIP Code:

#### LARGE TYPE WEEKLY

Developed especially for people with low vision, The New Herald Large Type Weekly offers a select package of the week's news printed in 16-point type--about twice the size of the regular type size. With its updated, color-enhanced design, The Large Type Weekly is a striking--and clearly readable way to enjoy The New Herald. A mail subscription of The New Herald Large Type Weekly makes a great gift for yourself or someone you care for.

To order, select a country/region below and submit.

Order Home Delivery at 50% Off (US Customers Only)

With convenient home delivery, you can start each day with all the news, the views, the wit and the wisdom you expect from The New Herald. And, through this special offer, you can save 50% on the first eight weeks when you order today. It's a smart, easy way to keep up with The Herald.

This offer expires December 31, 2001 and is valid in areas served by The New Herald' delivery service. Subscribers who have had Herald home delivery within the past 90 days are not eligible for this introductory offer. To subscribe enter your ZIP code below and submit.

**PATENT**  
**Docket No.: 4428-4001**

ZIP Code:

**GIFT SUBSCRIPTION OFFER**

There's no present like The Herald. And, when you order a gift subscription of 12-week home delivery of The New Herald, you'll save 50% on the regular rate. Hurry. This offer expires December 31, 2001. To order, enter the ZIP code of the gift recipient below and submit.

ZIP Code:

**LARGE TYPE WEEKLY**

Developed especially for people with low vision, The New Herald Large Type Weekly offers a select package of the week's news printed in 16-point type--about twice the size of the regular type size. With its updated, color-enhanced design, The Large Type Weekly is a striking--and clearly readable way to enjoy The New Herald. A mail subscription of The New Herald Large Type Weekly makes a great gift for yourself or someone you care for.

To order, select a country/region below and submit.

Country:

**The New Herald Book Review**

Get a head start on the latest book reviews, the acclaimed New Herald Best Sellers lists and everything new and noteworthy in the literary world. When you order a mail subscription to The New Herald Sunday Book Review, you'll receive it days in advance of the Sunday New Herald.

To order, select a country/region below and submit.

Country:

**To Subscribe - Foreign Mail Subscriptions**

Stay informed with all the news in the United States and throughout the world, including sharp analyses, reports and reviews from the world of business, sports and the arts. Discerning readers across the country and around the globe depend on The Herald for inside revelations, outside opinions, all sides of the story. Now you can too -- with the convenience of home delivery by mail. Order now.

To change your address, method of billing or any of the account information featured below, please enter the updated information in the appropriate box. Once you have completed all

**PATENT**  
**Docket No.: 4428-4001**

information requested, please click Submit at the bottom of the page. (Please note: It is necessary for all bold fields to be filled out to process your updated information.)

Your billing and payment history provides an at-a-glance summary of your account and makes it simple to check on your balance, last payment and new charges. Recent invoices are listed below for your review. Questions may be submitted to our customer care billing representatives by going to the Complaints page. Please be sure to indicate Billing Inquiry as the nature of your complaint.

Did you miss a paper? The New Herald is committed to making sure you get every issue you've ordered. If you did not receive your paper or any of its sections, simply select one of the following redelivery options so that we may deliver one to you.

Please note: Only same day redelivery is available on-line; you must submit this information on the same day on which your paper or section was to be delivered. For credits on past issues, please phone our customer care representatives at 1(800) 555-9876.

I would like to have today's paper delivered tomorrow. Please credit my account for today's missed paper.

Did you miss a paper? The New Herald is committed to making sure you get every issue you've ordered. If you did not receive your paper or any of its sections, simply select one of the following redelivery options so that we may deliver one to you.

Please note: Only same day redelivery is available on-line; you must submit this information on the same day on which your paper or section was to be delivered. For credits on past issues, please phone our customer care representatives at 1(800) 555-9876.

#### PAPER NOT RECEIVED

I did not receive today's paper.

ACCOUNT NUMBER: 060095544

Reason:

Select One: I would like to have today's paper delivered tomorrow.

Please credit my account for today's missed paper.

#### SECTIONS NOT RECEIVED

**PATENT**  
**Docket No.: 4428-4001**

I received today's paper with the section(s) checked below missing

Sections not received:      ARTS & LEISURE    BUSINESS  
   DINING IN/OUT    MAIN NEWS SEC  
   METROPOLITAN    SPORTS

Select One:    I would like to have today's paper delivered tomorrow.

Please credit my account for today's missed paper.

As a newspaper home delivery subscriber, you may suspend your service for any amount of time. When you suspend your home delivery service, you may elect to take part in our vacation donation program (see description below). Please indicate your suspension

and restart dates below:

ACCOUNT NUMBER: 060095544

SUSPEND/RESUME

Suspend:

Resume:

Vacation Donation Program

During your next vacation, sit back, relax -- and at the same time enrich your community. Through The New Herald Newspaper in Education program, you can donate your subscription to students for the time period in which you will be out of town. For each copy you donate, at least two students will receive their own copy of The New Herald. To donate your vacation copies, please indicate below.

Choose One:    Donate the vacation period papers to local schools through the Newspapers in Education program. Credit my account for the period of my vacation. SUSPEND/RESUME 2

Suspend:

Resume:



**PATENT**  
**Docket No.: 4428-4001**

Choose One: Donate the vacation period papers to local schools through the Newspapers in Education program. Credit my account for the period of my vacation. SUSPEND/RESUME 3

Suspend:

Resume:

Choose One: Donate the vacation period papers to local schools through the Newspapers in Education program. Credit my account for the period of my vacation.

To best provide you with responsive, accessible customer service, we encourage your comments and suggestions. Please let us know about any dissatisfaction you may have with your delivery or billing service. Customers who have not received a paper can order another paper or receive credit for today by clicking here.

For all other subscription concerns, please use the form below to send us an email indicating the nature of your complaint and explaining how we may help you. A customer care representative will respond to your request within 24 hours to the email address provided on this form.

s: Document 's'

---

a  
also  
an  
am  
and  
any  
as  
at  
be  
but  
by  
can  
could  
do  
for  
from  
go  
got  
have  
he  
her

**PATENT**  
**Docket No.: 4428-4001**

here  
his  
how  
i  
if  
in  
into  
it  
its  
m  
my  
of  
on  
or  
our  
say  
she  
that  
the  
their  
there  
therefore  
they  
there  
their  
this  
these  
those  
through  
to  
until  
we  
what  
when  
where  
which  
while  
who  
with  
would  
you  
your  
about

PATENT  
Docket No.: 4428-4001

across  
are  
around  
did  
during  
each  
ever  
every  
had  
must  
no  
now  
only  
other  
so  
than  
too  
was  
within  
everything  
is  
like  
want  
please  
s  
cannot  
then  
d  
ll  
me  
paper  
going  
having  
ve  
been  
being  
some  
speak  
know

f: Document 'f'

[FORM1]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="AcctNo":APrompt="Please tell us your account number.":Value="":RPrompt="Your account number is "  
 Type="AcceptResponse":Name="date":APrompt="When would you like to start suspending the paper?":Value="":RPrompt="The delivery will stop on "  
 Type="MChoice":Name="Duration":APrompt="Would you like to suspend the paper for one month, two months, or three months?":Choice="one month,two months,three months":Value="":RPrompt="The delivery will be suspended for "  
 Type="FormAction":Value="/susp\_deli"

[FORM2]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="Name":APrompt="Please tell us your name.":Value="":RPrompt="Your name is "  
 Type="AcceptResponse":Name="Address":APrompt="What city do you live in?":Value="":RPrompt="You live in "  
 Type="MChoice":Name="SubType":APrompt="Would you like the newspaper daily or just the Sunday paper?":Value="":Choice="a daily newspaper,the Sunday newspaper":RPrompt="You have opted for "  
 Type="MChoice":Name="SubPrd":APrompt="Would you like a half-yearly or annual subscription?":Value="":Choice="a half-yearly subscription,an annual subscription":RPrompt="You have chosen "  
 Type="FormAction":Value="/add\_acct"

[FORM3]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="AcctNo":APrompt="What is your account number?":Value="":RPrompt="Your account number is "  
 Type="FormAction":Value="/acct\_info"

[FORM4]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="AcctNo":APrompt="What is your account number?":Value="":RPrompt="Your account number is "  
 Type="FormAction":Value="/get\_pymt"

[FORM5]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="Name":APrompt="Whom would you like to gift the subscription to?":Value="":RPrompt="You are gifting this subscription to "

**PATENT**  
**Docket No.: 4428-4001**

Type="AcceptResponse":Name="Address":APrompt="In which city does the person live?":Value="":RPrompt="The person lives in "  
 Type="MChoice":Name="SubType":APrompt="Would you like to give a daily newspaper or just the Sunday paper?":Value="":Choice="a daily newspaper,the Sunday newspaper":RPrompt="You have opted for "  
 Type="MChoice":Name="SubPrd":APrompt="Would you like a half-yearly or annual subscription?":Value="":Choice="a half-yearly subscription,an annual subscription":RPrompt="You have chosen "  
 Type="FormAction":Value="/add\_acct"

## [FORM6]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="AcctNo":APrompt="What is your account number?":Value="":RPrompt="Your account number is "  
 Type="MChoice":Name="preference":APrompt="Would you like the newspaper or would you prefer credit for it?":Value="":Choice="the newspaper,credit":RPrompt="You prefer "  
 Type="FormAction":Value="/prefer"

## [FORM7]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="AcctNo":APrompt="What is your account number?":Value="":RPrompt="Your account number is "  
 Type="Hidden":Name="preference":Value="credit"  
 Type="FormAction":Value="/prefer"

## [FORM8]

Type="FormType":Value="AcceptForm"  
 Type="AcceptResponse":Name="AcctNo":APrompt="What is your account number?":Value="":RPrompt="Your account number is "  
 Type="FormAction":Value="/chg\_acct"

## x: Document 'x'

---

FORM1:Would you like to temporarily suspend your delivery?  
 FORM2:Would you like to order a subscription?  
 FORM3:Would you like information about your account balance or your payments?  
 FORM4:Would you like to pay your subscription fees?  
 FORM5:Would you like to give a gift subscription?  
 FORM6:Did you miss today's paper?  
 FORM6:Did you receive a wet paper?  
 FORM7:Did you miss yesterday's newspaper and would you like credit for yesterday?  
 FORM8:Would you like to change your address or change any other information?

**PATENT**  
**Docket No.: 4428-4001**

a: Datafile 'a' contains data about subscription  
 1|1|2|01-01-2002|365|315|01-01-2002|50|||Frege|Jena  
 2|2|2|01-02-2002|52|32|01-02-2002|20|||Russell|Cambridge  
 3|2|2|01-02-2002|52|32|01-02-2002|20|||Wittgenstein|Vienna  
 4|1|2|01-04-2002|364|314|01-04-2002|50|||Austin|Oxford  
 5|1|2|01-05-2002|365|264|01-05-2002|100|||Grice|Berkeley  
 6|1|1|01-06-2002|180|49|01-06-2002|130|||Parikh|New York

## CONFIGURATION FILES

**t.cfg:** Thesaurus configuration file. This is generated by program 't'

---

### [PROMPTS]

Are you calling about subscriptions?  
 Would you like to order a subscription?  
 Would you like to pay your subscription fees?  
 Would you like to give a gift subscription?  
 Would you like to change your address or change any other information?  
 Do you have any billing enquiries or concerns?  
 Would you like information about your account balance or your payments?  
 Would you like to speak to a customer care representative?  
 Would you like to temporarily suspend your delivery?  
 Is there a problem with your paper or delivery?  
 Did you miss today's paper?  
 Did you miss yesterday's newspaper and would you like credit for yesterday?  
 Did you receive a wet paper?  
 Would you like information about the New Herald website?  
 Would you like to obtain your New Herald website password?  
 The website address is [www.newherald.com](http://www.newherald.com). Would you like any other information about the website?  
 Are you calling about advertisements?  
 Would you like to advertise in the New Herald?  
 Is it a classified ad?  
 Is it a full-page, half-page, or quarter-page ad?  
 Would you like to place an ad?  
 Are you calling about something else?  
 Would you like to write to the New Herald?  
 Would you like to submit an article to the op-ed page?  
 Please email your article to [oped@newherald.com](mailto:oped@newherald.com).  
 Would you like to send a letter to the editor?  
 Please email your letter to [letters@newherald.com](mailto:letters@newherald.com).  
 Would you like to work for the New Herald?

**PATENT**  
**Docket No.: 4428-4001**

Would you like to write for the New Herald?  
Would you like to work for the editorial division or for the administrative division?

[MENUTREE]

0,1,0  
1,2,99  
1,3,99  
1,4,99  
1,5,99  
1,6,0  
6,7,99  
6,8,99  
1,9,99  
0,10,0  
10,11,99  
10,12,99  
10,13,99  
0,14,0  
14,15,99  
14,16,99  
0,17,0  
17,18,0  
18,19,99  
18,20,99  
17,21,0  
21,19,99  
21,20,99  
0,22,0  
22,23,0  
23,24,25  
24,25,100  
23,26,27  
26,27,100  
22,28,0  
28,29,99  
28,30,99

[INDEX]

account 7,  
ad 19,20,21,  
address 5,16,  
administr 30,  
advertis 17,18,

**PATENT**  
**Docket No.: 4428-4001**

annual 2,4,  
articl 24,25,  
balanc 7,  
bill 6,  
call 1,17,22,  
care 8,  
chang 5,  
classifi 19,  
com 16,25,27,  
concern 6,  
credit 12,11,13,  
custom 8,  
daili 2,4,  
deliveri 9,10,  
divis 30,  
ed 24,  
editor 26,  
editori 30,  
els 22,  
email 25,27,  
enquiri 6,  
fee 3,  
full 20,  
gift 4,  
give 4,  
half 20,2,4,  
inform 5,7,14,16,  
letter 26,27,  
miss 11,12,  
month 9,  
newspap 12,2,4,11,13,  
nytim 16,25,27,  
obtain 15,  
on 9,  
op 24,25,  
order 2,  
pai 3,  
password 15,  
payment 7,  
place 21,  
problem 10,  
quarter 20,  
receiv 13,



PATENT  
Docket No.: 4428-4001

repres 8,  
 send 26,  
 someth 22,  
 submit 24,  
 subscript 1,2,3,4,  
 sundai 2,4,  
 suspend 9,  
 temporarili 9,  
 three 9,  
 todai 11,  
 two 9,  
 websit 14,15,16,  
 wet 13,  
 work 28,30,  
 write 23,29,  
 www 16,  
 yearli 2,4,  
 yesterdai 12,

## [THESAURUS]

access 58,41,48,19,  
 acclaim 54,48,53,41,  
 account 16,36,39,34,  
 address 12,9,15,50,25,  
 advanc 54,48,53,41,  
 allow 11,17,9,32,  
 amount 55,36,19,  
 analys 19,41,32,52,  
 analysi 32,17,49,  
 anoth 58,41,48,19,  
 appropri 12,3,9,32,  
 ask 32,17,49,  
 back 59,48,36,53,  
 balanc 44,49,11,9,  
 benefit 9,12,17,19,  
 better 32,17,49,  
 bill 12,3,44,  
 bold 12,3,9,32,  
 bottom 12,3,9,32,  
 box 12,3,9,32,  
 brows 11,17,9,32,  
 care 49,17,8,32,  
 chang 9,3,44,19,

**PATENT**  
**Docket No.: 4428-4001**

charg 8,44,49,11,9,  
commun 59,48,36,53,  
complet 12,3,9,32,  
concern 50,25,49,11,3,  
consid 11,17,9,32,  
credit 1,36,34,39,58,  
custom 11,49,32,  
deliveri 41,52,  
depend 19,41,32,52,  
descript 55,36,19,  
discern 19,41,32,52,  
disclos 32,17,49,  
discov 11,17,9,32,  
dissatisfact 58,41,48,19,  
elect 55,36,19,  
email 15,50,49,3,11,17,  
encourag 58,41,48,19,  
enrich 59,48,36,53,  
explain 15,50,25,49,11,3,  
featur 12,3,9,32,  
field 12,3,9,32,  
fill 12,3,9,32,  
find 9,12,17,19,  
fit 11,17,9,32,  
foreign 53,29,59,54,  
gift 53,41,52,19,  
glanc 8,44,49,11,9,  
globe 19,41,32,52,  
guidelin 32,17,49,  
handl 53,29,59,54,  
head 54,48,53,41,  
highest 32,17,49,  
identifi 32,17,49,  
improv 32,17,49,  
individu 32,17,49,  
inform 17,49,11,  
inquiri 8,44,49,11,9,  
insid 19,41,32,52,  
instant 9,12,17,19,  
interest 32,17,49,  
invoic 8,44,49,11,9,  
last 8,44,49,11,9,  
latest 54,48,53,41,

**PATENT**  
**Docket No.: 4428-4001**

least 59,48,36,53,  
leisur 48,59,54,34,58,  
let 58,41,48,19,  
literari 54,48,53,41,  
manag 11,17,9,32,  
miss 58,16,39,1,48,  
most 53,29,59,54,  
necessari 12,3,9,32,  
newspap 16,55,39,1,  
next 59,48,36,53,  
noteworthi 54,48,53,41,  
notifi 11,17,9,32,  
on 36,16,34,1,58,  
onc 12,3,9,32,  
opinion 19,41,32,52,  
order 19,52,29,  
outsid 19,41,32,52,  
own 59,48,36,53,  
part 55,36,19,  
parti 32,17,49,  
payment 8,9,12,49,11,  
perform 32,17,49,  
person 32,17,49,  
pleas 9,12,17,19,  
polici 32,17,49,  
post 32,17,49,  
privaci 32,17,49,  
process 12,3,9,32,  
product 32,17,49,  
qualiti 32,17,49,  
receiv 34,59,  
recent 8,44,49,11,9,  
relax 59,48,36,53,  
repres 11,17,8,15,50,32,  
respond 15,50,25,49,11,3,  
revel 19,41,32,52,  
round 11,17,9,32,  
see 55,36,19,  
seller 54,48,53,41,  
send 15,25,49,11,3,  
sharp 19,41,32,52,  
short 32,17,49,  
side 19,41,32,52,



**PATENT**  
**Docket No.: 4428-4001**

```

cnt=`grep -c "^$acctno|" a`
if [ $cnt -eq 0 ]
then
    echo 'Type="FormType":Value="ResponseForm"'
    echo 'Type="Say":RPrompt="Sorry, the account number you provided does not
exist":Value=""'
    exit 0
fi
line=`grep "^$acctno|" a`
echo 'Type="FormType":Value="ResponseForm"'

name=`echo $line | cut -d'|' -f 11`
echo 'Type="Say":RPrompt="Your last name is ":Value=""$name"'
city=`echo $line | cut -d'|' -f 12`
echo 'Type="Say":RPrompt="You live in ":Value=""$city"'
sub_type=`echo $line | cut -d'|' -f 2`
if [ $sub_type -eq 1 ]
then
    sub_type="a daily newspaper"
else
    sub_type="the Sunday newspaper"
fi
echo 'Type="Say":RPrompt="You have subscribed for ":Value=""$sub_type"'
sub_prd=`echo $line | cut -d'|' -f 3`
sdate=`echo $line | cut -d'|' -f 4`
if [ $sub_prd -eq 1 ]
then
    sub_prd="six months"
else
    sub_prd="one year"
fi
echo 'Type="Say":RPrompt="The subscription starts on '$sdate' for a period of
":Value=""$sub_prd"'

fee=`echo $line | cut -d'|' -f 5`
echo 'Type="Say":RPrompt="The subscription fee is $":Value=""$fee"'
bal=`echo $line | cut -d'|' -f 6`
echo 'Type="Say":RPrompt="Your balance is $":Value=""$bal"'
pdate=`echo $line | cut -d'|' -f 7`
pymt=`echo $line | cut -d'|' -f 8`
echo 'Type="Say":RPrompt="Your last payment was '$pymt' on ":Value=""$pdate"'
sdate=`echo $line | cut -d'|' -f 9`
if [ "X$sdate" != "X" ]

```

PATENT  
Docket No.: 4428-4001

```

then
    suprd=`echo $line | cut -d'|' -f 10`
    case $suprd in
    1) suprd="one month";;
    2) suprd="two months";;
    3) suprd="three months";;
    esac
    echo "Type="Say":RPrompt="Your account is suspended from '$sdate' for
":Value=""$suprd""
fi

```

**add\_acct:** Script to add new account into 'a'

---

```

#!/bin/sh
# arrange all the values of input into a single line
cp /dev/null /tmp/param
cut -d=' ' -f 2 | sed "s/^//g
s/ ^\\\\\\\\\\\\/g" | while read aa
do
    echo -n $aa' ' >> /tmp/param
done
echo "" >>/tmp/param

# now transfer them into env variables.
read NAME CITY SUB_TYPE SUB_PRD < /tmp/param
if [ "$SUB_TYPE" = "a daily newspaper" ]
then
    SUB_TYPE=1
    FEE=182
else
    SUB_TYPE=2
    FEE=26
fi
if [ "$SUB_PRD" = "a half-yearly subscription" ]
then
    SUB_PRD=1
else
    SUB_PRD=2
    FEE=`expr $FEE \\* 2`
fi
cnt=1
while true

```

PATENT  
Docket No.: 4428-4001

```
do
  if [ "`grep -c \"^$cnt|\` a`" -ne 0 ]
  then
    cnt=`expr $cnt + 1`
    continue
  fi
  echo $cnt|$SUB_TYPE|$SUB_PRD|`date +%d-%m-%Y`|$FEE|$FEE|0||$NAME|$CITY" >> a
  echo 'Type="FormType":Value="AcceptForm"'
  echo 'Type="Say":RPrompt="Your subscription request has been entered":Value=""'
  echo 'Type="Say":Name="acct_no":RPrompt="Your account number is ":Value="$cnt"'
  echo 'Type="Say":RPrompt="Your fee for the subscription is $":Value="$FEE"'
  echo 'Type="AcceptResponse":Name="payment":APrompt="Your minimum initial
payment is $25. How much would you like to pay now?":Value="":RPrompt="You have chosen
to pay $"'
  echo 'Type="FormAction":Value="./updt_pymt"'
  break
done
#rm /tmp/param
```

**chg\_acct:** Script to generate a form to change account information

---

```
#!/bin/sh
acctno=`cut -d=' ' -f 2 | sed s/^\//g`
#echo $acctno
cnt=`grep -c "^$acctno|` a`
if [ $cnt -eq 0 ]
then
  echo 'Type="FormType":Value="ResponseForm"'
  echo 'Type="Say":RPrompt="Sorry, the account number you provided does not
exist":Value=""'
  exit 0
fi
line=`grep "^$acctno|` a`
echo 'Type="FormType":Value="AcceptForm"'
echo 'Type="Hidden":Name="acctno":Value="$acctno"'
#----- Response info
name=`echo $line | cut -d'|' -f 11`
echo 'Type="Say":RPrompt="Your last name is ":Value="$name"'
city=`echo $line | cut -d'|' -f 12`
echo 'Type="Say":RPrompt="You live in ":Value="$city"'
sub_type=`echo $line | cut -d'|' -f 2`
```

**PATENT**  
**Docket No.: 4428-4001**

```

if [ $sub_type -eq 1 ]
then
    sub_type="a daily newspaper"
else
    sub_type="the Sunday newspaper"
fi
echo 'Type="Say":RPrompt="You have subscribed for ":Value="$sub_type"'
sub_prd=`echo $line | cut -d'|' -f 3`
sdate=`echo $line | cut -d'|' -f 4`
if [ $sub_prd -eq 1 ]
then
    sub_prd="six months"
else
    sub_prd="one year"
fi
echo 'Type="Say":RPrompt="The subscription starts on '$sdate' for a period of
":Value="$sub_prd"'

fee=`echo $line | cut -d'|' -f 5`
echo 'Type="Say":RPrompt="The subscription fee is $":Value="$fee"'
bal=`echo $line | cut -d'|' -f 6`
echo 'Type="Say":RPrompt="Your balance is $":Value="$bal"'
pdate=`echo $line | cut -d'|' -f 7`
pymt=`echo $line | cut -d'|' -f 8`
echo 'Type="Say":RPrompt="Your last payment was $'$pymt' on ":Value="$pdate"'
sdate=`echo $line | cut -d'|' -f 9`
if [ "X$sdate" != "X" ]
then
    suprd=`echo $line | cut -d'|' -f 10`
    case $suprd in
    1) suprd="one month";;
    2) suprd="two months";;
    3) suprd="three months";;
    esac
    echo 'Type="Say":RPrompt="Your account is suspended from '$sdate' for
":Value="$suprd"'
fi
#-----
echo 'Type="AcceptResponse":Name="Name":APrompt="What name would you like to
use?":Value="":RPrompt="The name you would like to use is "'
echo 'Type="AcceptResponse":Name="Address":APrompt="What city would you like the
newspaper sent to?":Value="":RPrompt="The city you would like the newspaper sent to is "'

```



**PATENT**  
**Docket No.: 4428-4001**

```
echo 'Type="MChoice":Name="SubType":APrompt="Would you like the newspaper daily or
just the Sunday paper?":Value="":Choice="a daily newspaper,the Sunday
newspaper":RPrompt="You have opted for "'
echo 'Type="MChoice":Name="SubPrd":APrompt="Would you like a half-yearly or annual
subscription?":Value="":Choice="a half-yearly subscription,an annual
subscription":RPrompt="You have chosen "'
echo 'Type="FormAction":Value="/updt_acct"'
```

**get\_pymt:** Script to generate a form to accept payment for a particular account

---

```
#!/bin/sh
acctno=`cut -d=' ' -f 2 | sed s/^//g`
#echo $acctno
cnt=`grep -c "^$acctno|" a`
if [ $cnt -eq 0 ]
then
    echo 'Type="FormType":Value="ResponseForm"'
    echo 'Type="Say":RPrompt="Sorry, the account number you provided does not
exist":Value=""'
    exit 0
fi

line=`grep "^$acctno|" a`

fee=`echo $line | cut -d'|' -f 5`
bal=`echo $line | cut -d'|' -f 6`
pdate=`echo $line | cut -d'|' -f 7`
pymt=`echo $line | cut -d'|' -f 8`
if [ $bal -le 0 ]
then
    echo 'Type="FormType":Value="ResponseForm"'
else
    echo 'Type="FormType":Value="AcceptForm"'
fi
echo 'Type="Say":RPrompt="The subscription fee is $":Value="$fee"'
echo 'Type="Say":RPrompt="Your last payment was '$pymt' on":Value="$pdate"'
echo 'Type="Say":RPrompt="Your balance is $":Value="$bal"'
if [ $bal -ne 0 ]
then
    echo 'Type="Hidden":Name="acctno":Value="$acctno"'
    echo 'Type="AcceptResponse":Name="payment":APrompt="How much would you like
to pay now?":Value="":RPrompt="You have paid $"'
```



**PATENT**  
**Docket No.: 4428-4001**

```
if [ "$ans" = "yes" ]
then
    echo "Type="FormType":Value="ResponseForm"
    echo "Type="Say":RPrompt="Thank you for the subscription":Value=""
else
    echo "Type="FormType":Value="ResponseForm"
    echo "Type="Say":RPrompt="Sorry, the account number you provided does not
exist":Value=""
fi
rm /tmp/param1 /tmp/found
```

**susp\_deli:** Script to suspend delivery for a particular account

---

```
#!/bin/sh
cp /dev/null /tmp/param1
cut -d=' ' -f 2 | sed "s/^//g"
s/ ^\\\\\\\\\\\\/g" | while read aa
do
    echo -n $aa' ' >> /tmp/param1
done
echo "" >>/tmp/param1
read acctno sdate period < /tmp/param1
if [ "$period" = "one month" ]
then
    period=1
fi
if [ "$period" = "two month" ]
then
    period=2
fi
if [ "$period" = "three months" ]
then
    period=3
fi
echo "no" > /tmp/found
touch /tmp/tmpa
cat a | while read line
do
    cacno=`echo $line | cut -d'|' -f 1`
    if [ $cacno -eq $acctno ]
    then
        echo "yes" > /tmp/found
```

**PATENT**  
**Docket No.: 4428-4001**

```

echo -n $cacno'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 2`'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 3`'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 4`'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 5`'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 6`'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 7`'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 8`'| >> /tmp/tmpa
echo -n $sdate'| >> /tmp/tmpa
echo -n $period'| >> /tmp/tmpa
echo -n `echo $line | cut -d'|' -f 11`'| >> /tmp/tmpa
echo `echo $line | cut -d'|' -f 12` >> /tmp/tmpa
else
    echo $line >> /tmp/tmpa
fi
done
mv /tmp/tmpa a
read ans < /tmp/found
if [ "$ans" = "yes" ]
then
    echo "Type="FormType":Value="ResponseForm"
    echo "Type="Say":RPrompt="Thank you. The information has been updated":Value=""
else
    echo "Type="FormType":Value="ResponseForm"
    echo "Type="Say":RPrompt="Sorry, the account number you provided does not
exist":Value=""
fi

rm /tmp/param1 /tmp/found

```

**updt\_acct:** Script to update data file 'a' with changed information

```

#!/bin/sh
# arrange all the values of input into a single line
cp /dev/null /tmp/param
cut -d=' ' -f 2 | sed "s^//g
s/ ^\\\\\\\\\\\\/g" | while read aa
do
    echo -n $aa' ' >> /tmp/param
done
echo "" >>/tmp/param
cp /dev/null /tmp/tmpa

```

PATENT  
Docket No.: 4428-4001

```

read acctno name city sub_type sub_prd < /tmp/param
cat a | while read line
do
  cacno=`echo $line | cut -d'|' -f 1`
  if [ $cacno -eq $acctno ]
  then
    echo -n $cacno'|' >> /tmp/tmpa
    if [ "$sub_type" = "a daily newspaper" ]
    then
      sub_type=1
      newfee=182
    else
      sub_type=2
      newfee=26
    fi
    echo -n $sub_type'|' >> /tmp/tmpa
    if [ "$sub_prd" = "a half-yearly subscription" ]
    then
      sub_prd=1
    else
      sub_prd=2
      newfee=`expr $newfee \* 2`
    fi
    echo -n $sub_prd'|' >> /tmp/tmpa
    echo -n `echo $line | cut -d'|' -f 4`'|' >> /tmp/tmpa
    echo -n $newfee'|' >> /tmp/tmpa
    oldfee=`echo $line | cut -d'|' -f 5`
    oldbal=`echo $line | cut -d'|' -f 6`
    newbal=`expr $newfee - $oldfee + $oldbal`
    echo $newfee '$newbal' > /tmp/newbal
    if [ $newbal -gt 0 ]
    then
      echo -n $newbal'|' >> /tmp/tmpa
    else
      echo -n '0'|' >> /tmp/tmpa
    fi
    #echo -n `date +%d-%m-%Y`'|' >> /tmp/tmpa
    echo -n `echo $line | cut -d'|' -f 7`'|' >> /tmp/tmpa
    echo -n `echo $line | cut -d'|' -f 8`'|' >> /tmp/tmpa
    echo -n `echo $line | cut -d'|' -f 9`'|' >> /tmp/tmpa
    echo -n `echo $line | cut -d'|' -f 10`'|' >> /tmp/tmpa
    echo -n $name'|' >> /tmp/tmpa
    echo $city >> /tmp/tmpa

```

```

else
    echo $line >> /tmp/tmpa
fi
done
mv /tmp/tmpa a
echo "Type="FormType":Value="ResponseForm"
read newfee newbal < /tmp/newbal
echo "Type="Say":RPrompt="Your fee for the subscription is $":Value="$newfee"
if [ $newbal -lt 0 ]
then
    newbal=`expr $newbal \* -1`
    echo "Type="Say":RPrompt="A cheque of '$newbal' will be sent to you to compensate
for excess balance"
else
    echo "Type="Say":RPrompt="Your balance is $":Value="$newbal"
fi
echo "Type="Say":RPrompt="Thank you":Value=""
rm /tmp/param /tmp/newbal

```

**prefer:** Script to generate form for damaged / missing newspaper complaint

---

```

#!/bin/sh
cp /dev/null /tmp/param1
cut -d=' ' -f 2 | sed "s/^//g"
s/ ^\\\\\\\\/g" | while read aa
do
    echo -n $aa' ' >> /tmp/param1
done
echo "" >>/tmp/param1
read acctno preference < /tmp/param1
cnt=`grep -c "^$acctno|" a`
if [ $cnt -eq 0 ]
then
    echo "Type="FormType":Value="ResponseForm"
    echo "Type="Say":RPrompt="Sorry, the account number you provided does not
exist":Value=""
    exit 0
fi

if [ "$preference" = "the newspaper" ]
then
    echo "Type="FormType":Value="ResponseForm"

```

**PATENT**  
**Docket No.: 4428-4001**

```

echo 'Type="Say":RPrompt="You will be sent today\'s newspaper":Value=""'
echo 'Type="Say":RPrompt="Thank you":Value=""'
exit 0
fi
touch /tmp/tmpa
cat a | while read line
do
    cacno=`echo $line | cut -d'|' -f 1`
    if [ $cacno -eq $acctno ]
    then
        echo "yes" > /tmp/found
        echo -n $cacno'|' >> /tmp/tmpa
        echo -n `echo $line | cut -d'|' -f 2`'|' >> /tmp/tmpa
        echo -n `echo $line | cut -d'|' -f 3`'|' >> /tmp/tmpa
        echo -n `echo $line | cut -d'|' -f 4`'|' >> /tmp/tmpa
        echo -n `echo $line | cut -d'|' -f 5`'|' >> /tmp/tmpa
        bal=`echo $line | cut -d'|' -f 6`
        bal=`expr $bal - 1`
        echo -n "$bal"|' >> /tmp/tmpa
        echo -n `echo $line | cut -d'|' -f 7`'|' >> /tmp/tmpa
        echo -n `echo $line | cut -d'|' -f 8`'|' >> /tmp/tmpa
        echo -n $sdate'|' >> /tmp/tmpa
        echo $period >> /tmp/tmpa
    else
        echo $line >> /tmp/tmpa
    fi
done
mv /tmp/tmpa a
echo 'Type="FormType":Value="ResponseForm"'
echo 'Type="Say":RPrompt="Your account has been credited":Value=""'
echo 'Type="Say":RPrompt="Thank you":Value=""'
rm /tmp/param1 /tmp/found

```

## WE CLAIM:

SUB A17

1. A method performed in a system having multiple navigable nodes interconnected in a hierarchical arrangement comprising:  
at a first node, receiving an input from a user of the system, the input containing at least one word identifiable with at least one keyword from among multiple keywords,  
identifying at least one node, other than the first node, that is not directly connected to the first node but is associated with the at least one keyword, and  
jumping to the at least one node.
2. The method of claim 1 further comprising:  
providing a verbal description associated with the at least one node to the user.
3. The method of claim 1 further comprising:  
searching a thesaurus correlating keywords with synonyms.
4. The method of claim 3 wherein the searching further comprises:  
identifying the at least one word as synonymous with the at least one keyword.
5. The method of claim 1 further comprising:  
determining that the at least one word is neither a keyword nor a synonym of any  
keyword; and



**PATENT**  
**Docket No.: 4428-4001**

learning a meaning for the word so that the word will be treated as a learned synonym for at least one particular keyword of the multiple keywords.

6. The method of claim 5 further comprising:  
adding the word to a thesaurus so that, when the word is input by a subsequent user, the word will be treated as synonymous with the at least one particular keyword.

7. A method performed in connection with an arrangement of nodes representable as a hierarchical graph containing vertices and edges connecting at least two of the vertices, the method comprising:

receiving an input from a user as a response to a verbal description associated with a first vertex;

analyzing the input to identify a meaningful term that can be associated with at least one keyword;

selecting a vertex in the graph structure that is not connected by an edge to the first vertex, based upon an association between the meaningful term and the at least one keyword and a correlation between the at least one keyword and the vertex; and

jumping to the vertex.

**PATENT**  
**Docket No.: 4428-4001**

8. A method performed in connection with an arrangement of nodes representable as a hierarchical graph comprising:

correlating keywords with nodes in which the keywords appear to create an inverted index so that the keywords each appear only once and all nodes containing each of the keywords are indexed to those keywords;

maintaining a thesaurus of synonyms for at least some of the keywords;

receiving an input from a user containing a meaningful word;

searching the inverted index to determine whether the meaningful word is a keyword and, if the meaningful word is a keyword, jumping to a node identified in the inverted index as correlated to that keyword, otherwise,

searching the thesaurus to determine if the meaningful word is a synonym for at least one particular keyword and, if the meaningful word is the synonym, using the synonym to identify the at least one particular keyword, and

jumping to at least one node correlated to the at least one particular keyword.

9. The method of claim 8 further comprising:

creating the thesaurus by analyzing at least two files and determining synonymy among application meaningful words contained therein based upon a frequency of co-occurrence among the application meaningful words.

10. A system comprising:  
a hierarchically arranged series of nodes;  
an inverted index correlating keywords with the nodes;  
a thesaurus correlating at least some keywords with synonyms for those keywords;  
a processor executable learning procedure configured to, upon receipt of a term that is identified as neither a synonym nor a keyword based upon a search of both the inverted index and the thesaurus,

(a) identify the term as at least one particular synonym for at least one particular keyword and

(b) correlate the term with the at least one particular keyword,  
so that when a subsequent user provides the term the system will operate as if the term was synonymous with the at least one particular keyword.

11. The system of claim 10 further comprising:

a set of verbal descriptions for at least some of the nodes.

12. The system of claim 10 wherein at least one of the nodes is a service node.

13. The system of claim 10 further comprising an interactive voice response system and wherein the hierarchically arranged series of nodes is part of the interactive voice response system.

**PATENT**  
**Docket No.: 4428-4001**

14. The system of claim 10 wherein the hierarchically arranged series of nodes is part of a file system browser application.

15. The system of claim 10 wherein the hierarchically arranged series of nodes is part of a navigation system for television listings.

16. The system of claim 10 wherein the hierarchically arranged series of nodes is part of one of a document navigation or a document retrieval system.

17. The system of claim 10 wherein the hierarchically arranged series of nodes is part of a geographic information system.

18. A transaction processing system, having a hierarchical arrangement of nodes and configured to interact with a user so that the user can navigate among the nodes in the hierarchy, the system comprising:

an inverted index correlating keywords with at least some of the nodes in the hierarchical arrangement so that when the user interacts with the system and provides an input in response to a verbal description from one node in the hierarchy and the response includes a meaningful word correlatable with a keyword, the system will identify at least one node that is correlated to the meaningful word by the inverted index and jump to that at least one node without first traversing any other node.

19. The system of claim 18 further comprising:  
a thesaurus correlating at least some of the keywords with synonyms for the at least some  
keywords.

20. The system of claim 18 further comprising:  
at least one stored learned word correlated to a keyword.

21. A method performed by a program executed by a processor to navigate among a  
hierarchically arranged group of nodes, each of the nodes having an associated verbal  
description, the method comprising:

eliminating stop words and duplicates from the verbal descriptions to create a list of  
keywords;

creating a list of thesaurus words;

creating a first matrix comprising a correlation of at least some thesaurus words with at  
least some keywords;

creating a second matrix by calculating cosine values from a co-occurrence analysis of  
the entries in the first matrix;

determining a synonymy among the at least some thesaurus words and the at least some  
keywords; and

creating a thesaurus configured as an inverted index based upon the synonymy.

**PATENT**  
**Docket No.: 4428-4001**

22. The method of claim 21 further comprising:  
tracking frequency of use of the nodes.
23. The method of claim 22 further comprising:  
ranking the nodes based upon a result of the tracking.
24. The method of claim 21 further comprising:  
pruning a node from the group of nodes based upon a frequency of usage criterion.
25. The method of claim 21 further comprising:  
adding a synonym entry into the thesaurus based upon a result of an unknown word  
analysis.
26. The method of claim 21 wherein the thesaurus further comprises at least some  
learned entries, the method further comprising:  
deleting a learned entry based upon satisfaction of a frequency of use criterion.

**PATENT**  
**Docket No.: 4428-4001**

### **ABSTRACT**

A method performed in a system having multiple navigable nodes interconnected in a hierarchical arrangement involves receiving an input containing at least one word identifiable with at least one keyword, identifying at least one node, other than the first node, not directly connected to the first node, but associated with the at least one keyword, and jumping to the identified node. A transaction processing system having a hierarchical arrangement of nodes and is configured for user navigation among the nodes. The system has an inverted index correlating keywords with at least some nodes in the arrangement so that when the user provides an input in response to a verbal description and the response includes a meaningful word correlatable with a keyword, the system will identify at least one node correlated to the meaningful word by the inverted index and jump to that node without first traversing any other node.

**COMBINED DECLARATION AND POWER OF ATTORNEY FOR  
ORIGINAL, DESIGN, NATIONAL STAGE OF PCT, SUPPLEMENTAL,  
DIVISIONAL, CONTINUATION OR CONTINUATION-IN-PART APPLICATION**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

**NAVIGATION IN A HIERARCHICAL  
STRUCTURED TRANSACTION PROCESSING SYSTEM**

the specification of which

- a.  is attached hereto.
- b.  was filed on \_\_\_\_\_ as application Serial No. \_\_\_\_\_ and was amended on \_\_\_\_\_ (if applicable).

**PCT FILED APPLICATION ENTERING NATIONAL STAGE**

- c.  was described and claimed in International Application No. \_\_\_\_ filed on \_\_\_\_ and as amended on \_\_\_\_ (if any).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 C.F.R. § 1.56.

I hereby specify the following as the correspondence address to which all communications about this application are to be directed:

**SEND CORRESPONDENCE TO:**

- Bar Code label attached (see right)
- Address Shown (see below)

MORGAN & FINNEGAN, L.L.P.  
345 Park Avenue  
New York, N.Y. 10154



27123

PATENT TRADEMARK OFFICE

↑AFFIX CUSTOMER NO. LABEL ABOVE ↑

**DIRECT TELEPHONE CALLS TO:** 212-758-4800



- I hereby claim foreign priority benefits under Title 35, United States Code § 119 (a)-(d) or under § 365(b) of any foreign application(s) for patent or inventor's certificate or under § 365(a) of any PCT international application(s) designating at least one country other than the U.S. listed below and also have identified below such foreign application(s) for patent or inventor's certificate or such PCT international application(s) filed by me on the same subject matter having a filing date within twelve (12) months before that of the application on which priority is claimed:
- The attached 35 U.S.C. § 119 claim for priority for the application(s) listed below forms a part of this declaration.

Country/PCT	Application Number	Date of filing (day, month, yr)	Date of issue (day, month, yr)	Priority Claimed
				<input type="checkbox"/> Y <input type="checkbox"/> N
				<input type="checkbox"/> Y <input type="checkbox"/> N
				<input type="checkbox"/> Y <input type="checkbox"/> N

- I hereby claim the benefit under 35 U.S.C. § 119(e) of any U.S. provisional application(s) listed below.

Provisional Application No.	Date of filing (day, month, yr)

**ADDITIONAL STATEMENTS FOR DIVISIONAL, CONTINUATION OR CONTINUATION-IN-PART OR PCT INTERNATIONAL APPLICATION(S) DESIGNATING THE U.S.)**

I hereby claim the benefit under Title 35, United States Code § 120 of any United States application(s) or under § 365(c) of any PCT international application(s) designating the U.S. listed below.

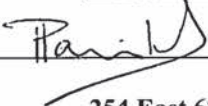
US/PCT Application Serial No.	Filing Date	Status (patented, pending, abandoned)/ U.S. application no. assigned (For PCT)

- In this continuation-in-part application, insofar as the subject matter of any of the claims of this application is not disclosed in the above listed prior United States or PCT international application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56(a) which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or Imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

I hereby appoint the following attorneys and/or agents with full power of substitution and revocation, to prosecute this application, to receive the patent, and to transact all business in the Patent and Trademark Office connected therewith: David H. Pfeffer (Reg. No. 19,825), Harry C. Marcus (Reg. No. 22,390), Robert E. Paulson (Reg. No. 21,046), Stephen R. Smith (Reg. No. 22,615), Kurt E. Richter (Reg. No. 24,052), J. Robert Dailey (Reg. No. 27,434), Eugene Moroz (Reg. No. 25,237), John F. Sweeney (Reg. No. 27,471), Arnold I. Rady (Reg. No. 26,601), Christopher A. Hughes (Reg. No. 26,914), William S. Feiler (Reg. No. 26,728), Joseph A. Calvaruso (Reg. No. 28,287), James W. Gould (Reg. No. 28,859), Richard C. Komson (Reg. No. 27,913), Israel Blum (Reg. No. 26,710), Bartholomew Verdirame (Reg. No. 28,483), Maria C.H. Lin (Reg. No. 29,323), Joseph A. DeGirolamo (Reg. No. 28,595), Michael P. Dougherty (Reg. No. 32,730), Seth J. Atlas (Reg. No. 32,454), Andrew M. Riddles (Reg. No. 31,657), Bruce D. DeRenzi (Reg. No. 33,676), Mark J. Abate (Reg. No. 32,527), John T. Gallagher (Reg. No. 35,516), Steven F. Meyer (Reg. No. 35,613), Kenneth H. Sonnenfeld (Reg. No. 33,285), Tony V. Pezzano (Reg. No. 38,271), Andrea L. Wayda (Reg. 43,979), Walter G. Hanchuk (Reg. No. 35,179), John W. Osborne (Reg. No. 36,231), Robert K. Goethals (Reg. No. 36,813), Peter N. Fill (Reg. No. 38,876), Mary J. Morry (Reg. No. 34,398) and Kenneth S. Weitzman (Reg. No. 36,306) of Morgan & Finnegan, L.L.P. whose address is: 345 Park Avenue, New York, New York, 10154; and Michael S. Marcus (Reg. No. 31,727), and John E. Hoel (Reg. No. 26,279), of Morgan & Finnegan, L.L.P., whose address is 1775 Eye Street, Suite 400, Washington, D.C. 20006.

- I hereby authorize the U.S. attorneys and/or agents named hereinabove to accept and follow instructions from us as to any action to be taken in the U.S. Patent and Trademark Office regarding this application without direct communication between the U.S. attorneys and/or agents and me. In the event of a change in the person(s) from whom instructions may be taken I will so notify the U.S. attorneys and/or agents named hereinabove.

Full name of second inventor:	<u>PRASHANT PARIKH</u>	
Inventor's signature*		<u>18 Nov. 2002</u> Date
Residence:	<u>254 East 68th Street, Apart. 21D, New York, New York 10021</u>	
Citizenship:	<u>Indian</u>	
Post Office Address:	<u>Same as residence</u>	

Full name of second inventor:	<u>STANLEY PETERS</u>	
Inventor's signature*	<u><i>Stanley Peters</i></u>	<u>13 Nov. 2002</u> Date
Residence:	<u>128 Hillside Avenue, Menlo Park, CA 94025</u>	
Citizenship:	<u>U.S.A.</u>	
Post Office Address:	<u>Same as residence</u>	

ATTACHED IS ADDED PAGE TO COMBINED DECLARATION AND POWER OF ATTORNEY FOR SIGNATURE BY THIRD AND SUBSEQUENT INVENTORS FORM.

\*Before signing this declaration, each person signing must:

1. Review the declaration and verify the correctness of all information therein; and
2. Review the specification and the claims, including any amendments made to the claims.

After the declaration is signed, the specification and claims are not to be altered.

To the inventor(s):

The following are cited in or pertinent to the declaration attached to the accompanying application:

Title 37, Code of Federal Regulation, §1.56

Duty to disclose information material to patentability

- (a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§ 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

- (1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

- (2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.
- (b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made of record in the application, and
- (1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or
- (2) It refutes, or is inconsistent with, a position the applicant takes in:
- (i) Opposing an argument of unpatentability relied on by the Office, or
- (ii) Asserting an argument of patentability.
- (iii) A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.
- (c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:
- (1) Each inventor named in the application;
- (2) Each attorney or agent who prepares or prosecutes the application; and
- (3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.
- (d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.
- (e) In any continuation-in-part application, the duty under this section includes the duty to disclose to the Office all information known to the person to be material to patentability, as defined in paragraph (b) of this section, which became available between the filing date of the prior application and the National or PCT international filing date of the continuation-in-part application.

#### Title 35, U.S. Code § 101

##### Inventions patentable

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

#### Title 35 U.S. Code § 102

##### Conditions for patentability; novelty and loss of right to patent

A person shall be entitled to a patent unless --

- (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for patent, or
- (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States, or
- (c) The has abandoned the invention, or
- (d) the invention was first patented or caused to be patented, or was the subject of an inventor's certificate, by the applicant or his legal representatives or assigns in a foreign country prior to the date of the application for patent in this country on an application for patent or inventor's certificate filed more than twelve months before the filing of the application in the United States, or
- (e) The invention was described in--
  - (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effect under this subsection of a national application published under section 122(b) only if the international application designating the United States was published under Article 21(2)(a) of such treaty in the English language; or
  - (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that a patent shall not be deemed filed in the United States for the purposes of this subsection based on the filing of an international application filed under the treaty defined in section 351(a); or
- (f) he did not himself invent the subject matter sought to be patented, or
- (g) (1) during the course of an interference conducted under section 135 or section 291, another inventor involved therein establishes, to the extent permitted in section 104, that before such person's invention thereof the invention was made by such other inventor and not abandoned, suppressed, or concealed, or (2) before such person's invention thereof, the invention was made in this country by another inventor who had not abandoned, suppressed, or concealed it. In determining priority of invention under this subsection, there shall be considered not only the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice, from a time prior to conception by the other.

Title 35, U.S. Code § 103

### 103. Conditions for patentability; non obvious subject matter

- (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.
- (b) (1) Notwithstanding subsection (a), and upon timely election by the applicant for patent to proceed under this subsection, a biotechnological process using or resulting in a composition of matter that is novel under section 102 and nonobvious under subsection (a) of this section shall be considered nonobvious if—

- (A) claims to the process and the composition of matter are contained in either the same application for patent or in separate applications having the same effective filing date; and
  - (B) the composition of matter, and the process at the time it was invented, were owned by the same person or subject to an obligation of assignment to the same person.
- (2) A patent issued on a process under paragraph (1)—
- (A) shall also contain the claims to the composition of matter used in or made by that process, or
  - (B) shall, if such composition of matter is claimed in another patent, be set to expire on the same date as such other patent, notwithstanding section 154.
- (3) For purposes of paragraph (1), the term "biotechnological process" means--
- (A) a process of genetically altering or otherwise inducing a single- or multi-celled organism to--
    - (i) express an exogenous nucleotide sequence,
    - (ii) inhibit, eliminate, augment, or alter expression of an endogenous nucleotide sequence, or
    - (iii) express a specific physiological characteristic not naturally associated with said organism;
  - (B) cell fusion procedures yielding a cell line that expresses a specific protein, such as a monoclonal antibody; and
  - (C) a method of using a product produced by a process defined by subparagraph (A) or (B), or a combination of subparagraphs (A) and (B).
- (c) Subject matter developed by another person, which qualifies as prior art only under one or more of subsections (e), (f), and (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person.

Title 35, U.S. Code § 112 (in part)

#### Specification

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention.

The specification shall conclude with one or more claims particularly printing out and distinctly claiming the subject matter which the applicant regards as his invention.

Title 35, U.S. Code, § 119

#### Benefit of earlier filing date; right of priority

- (a) An application for patent for an invention filed in this country by any person who has, or whose legal representatives or assigns have, previously regularly filed an application for a patent for the same invention in a foreign country which affords similar privileges in the case of applications filed in the United States or to citizens of the United States, or in a WTO member country, shall have the same effect as the same

application would have if filed in this country on the date on which the application for patent for the same invention was first filed in such foreign country, if the application in this country is filed within twelve months from the earliest date on which such foreign application was filed; but no patent shall be granted on any application for patent for an invention which had been patented or described in a printed publication in any country more than one year before the date of the actual filing of the application in this country, or which had been in public use or on sale in this country more than one year prior to such filing.

- (b)
- (1) No application for patent shall be entitled to this right of priority unless a claim is filed in the Patent and Trademark Office, identifying the foreign application by specifying the application number on that foreign application, the intellectual property authority or country in or for which the application was filed, and the date of filing the application, at such time during the pendency of the application as required by the Director.
  - (2) The Director may consider the failure of the applicant to file a timely claim for priority as a waiver of any such claim. The Director may establish procedures, including the payment of a surcharge, to accept an unintentionally delayed claim under this section.
  - (3) The Director may require a certified copy of the original foreign application, specification, and drawings upon which it is based, a translation if not in the English language, and such other information as the Director considers necessary. Any such certification shall be made by the foreign intellectual property authority in which the foreign application was filed and show the date of the application and of the filing of the specification and other papers.
- (c) In like manner and subject to the same conditions and requirements, the right provided in this section may be based upon a subsequent regularly filed application in the same foreign country instead of the first filed foreign application, provided that any foreign application filed prior to such subsequent application has been withdrawn, abandoned, or otherwise disposed of, without having been laid open to public inspection and without leaving any rights outstanding, and has not served, nor thereafter shall serve, as a basis for claiming a right of priority.
- (d) Applications for inventors' certificates filed in a foreign country in which applicants have a right to apply, at their discretion, either for a patent or for an inventor's certificate shall be treated in this country in the same manner and have the same effect for purpose of the right of priority under this section as applications for patents, subject to the same conditions and requirements of this section as apply to applications for patents, provided such applicants are entitled to the benefits of the Stockholm Revision of the Paris Convention at the time of such filing.
- (e)
- (1) An application for patent filed under section 111(a) or section 363 of this title for an invention disclosed in the manner provided by the first paragraph of section 112 of this title in a provisional application filed under section 111(b) of this title, by an inventor or inventors named in the provisional application, shall have the same effect, as to such invention, as though filed on the date of the provisional application filed under section 111(b) of this title, if the application for patent filed under section 111(a) or section 363 of this title is filed not later than 12 months after the date on which the provisional application was filed and if it contains or is amended to contain a specific reference to the provisional application. No application shall be entitled to the benefit of an earlier filed provisional application under this subsection unless an amendment containing the specific reference to the earlier filed provisional application is submitted at such time during the pendency of the application as required by the Director. The Director may consider the failure to submit such an amendment within that time period as a waiver of any benefit under this subsection. The Director may establish procedures, including the payment of a surcharge, to accept an unintentionally delayed submission of an amendment under this subsection during the pendency of the application.
  - (2) A provisional application filed under section 111(b) of this title may not be relied upon in any

proceeding in the Patent and Trademark Office unless the fee set forth in subparagraph (A) or (C) of section 41(a)(1) of this title has been paid.

- (3) If the day that is 12 months after the filing date of a provisional application falls on a Saturday, Sunday, or Federal holiday within the District of Columbia, the period of pendency of the provisional application shall be extended to the next succeeding secular or business day.
- (f) Applications for plant breeder's rights filed in a WTO member country (or in a foreign UPOV Contracting Party) shall have the same effect for the purpose of the right of priority under subsections (a) through (c) of this section as applications for patents, subject to the same conditions and requirements of this section as apply to applications for patents.
- (g) As used in this section--
  - (1) the term "WTO member country" has the same meaning as the term is defined in section 104(b)(2) of this title; and
  - (2) the term "UPOV Contracting Party" means a member of the International Convention for the Protection of New Varieties of Plants.

Title 35, U.S. Code, § 120

#### Benefit of earlier filing date in the United States

An application for patent for an invention disclosed in the manner provided by the first paragraph of section 112 of this title in an application previously filed in the United States, or as provided by section 363 of this title, which is filed by an inventor or inventors named in the previously filed application shall have the same effect, as to such invention, as though filed on the date of the prior application, if filed before the patenting or abandonment of or termination of proceedings on the first application or on an application similarly entitled to the benefit of the filing date of the first application and if it contains or is amended to contain a specific reference to the earlier filed application. No application shall be entitled to the benefit of an earlier filed application under this section unless an amendment containing the specific reference to the earlier filed application is submitted at such time during the pendency of the application as required by the Director. The Director may consider the failure to submit such an amendment within that time period as a waiver of any benefit under this section. The Director may establish procedures, including the payment of a surcharge, to accept an unintentionally delayed submission of an amendment under this section.

Please read carefully before signing the Declaration attached to the accompanying Application. If you have any questions, please contact Morgan & Finnegan, L.L.P.



PATENT

Docket No. 4428-4001

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Prashant Parikh  
Stanley Peters

Serial No.: To Be Assigned                      Examiner: To Be Assigned

Filed: Herewith                                      Group Art Unit: To Be Assigned

For: NAVIGATION IN A HIERARCHICAL  
STRUCTURED TRANSACTION PROCESSING SYSTEM

Commissioner Of Patents  
Washington, D.C. 20231

ASSOCIATE POWER OF ATTORNEY (37 C.F.R. 1.34)

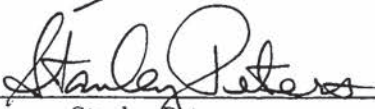
Please recognize as an Associate Practitioner in this case:

Richard Straussman  
Morgan & Finnegan, LLP  
345 Park Avenue  
New York, NY 10154  
Reg. No.: 39,847


Date: 18 Nov. 2002

Signed:   
Prashant Parikh

Date: 13 Nov. 2002

Signed:   
Stanley Peters

Date: 18 Nov. 2002

**SEMIOSIS L.L.C.**  
Signed:   
Name (Print): PRASHANT PARIKH  
Its (Title): CHAIRMAN + CEO

Correspondence Address:  
Morgan & Finnegan, LLP  
345 Park Avenue  
New York, NY 10154  
Tel.:(212)758-4800/Fax: (212)751-6849

PATENT APPLICATION SERIAL NO. \_\_\_\_\_

U.S. DEPARTMENT OF COMMERCE  
PATENT AND TRADEMARK OFFICE  
FEE RECORD SHEET

11/21/2002 DEMMANU1 00000082 10299359

01 FC:2001	370.00	OP
02 FC:2201	126.00	OP
03 FC:2202	54.00	OP

PTO-1556  
(5/87)

**PATENT APPLICATION FEE DETERMINATION RECORD**  
Effective October 1, 2001

Application or Docket Number

4428-4001

**CLAIMS AS FILED - PART I**

(Column 1) (Column 2)

TOTAL CLAIMS	26	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	26 minus 20 = *	6
INDEPENDENT CLAIMS	6 minus 3 = *	3
MULTIPLE DEPENDENT CLAIM PRESENT <input type="checkbox"/>		

\* If the difference in column 1 is less than zero, enter "0" in column 2

**SMALL ENTITY TYPE**  OR

**OTHER THAN SMALL ENTITY**

RATE	FEE
BASIC FEE	370.00
X\$ 9=	54
X42=	126
+140=	
TOTAL	550

RATE	FEE
BASIC FEE	740.00
X\$18=	
X84=	
+280=	
TOTAL	

**CLAIMS AS AMENDED - PART II**

(Column 1) (Column 2) (Column 3)

AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	Minus **	=
	Independent	*	Minus ***	=
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

**SMALL ENTITY** OR

**OTHER THAN SMALL ENTITY**

RATE	ADDITIONAL FEE
X\$ 9=	
X42=	
+140=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X84=	
+280=	
TOTAL ADDIT. FEE	

(Column 1) (Column 2) (Column 3)

AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	Minus **	=
	Independent	*	Minus ***	=
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

RATE	ADDITIONAL FEE
X\$ 9=	
X42=	
+140=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X84=	
+280=	
TOTAL ADDIT. FEE	

(Column 1) (Column 2) (Column 3)

AMENDMENT C	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	Minus **	=
	Independent	*	Minus ***	=
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

RATE	ADDITIONAL FEE
X\$ 9=	
X42=	
+140=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X84=	
+280=	
TOTAL ADDIT. FEE	

\* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."

\*\*\* If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.