YILDIZ TECHNICAL UNIVERSITY
ELECTRICAL- ELECTRONICS
FACULTY
COMPUTER ENGINEERING
DEPARTMENT

Algorithm Analysis Lecture
Third Assignment

KAMRAN BALAYEV 17011904

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


struct row {
    char *slct;
    char *input;
};

void getSelectedWord( char * userInput ,char * selectedInput,int selectedHash, struct row *errorTable,char *hashTable);

int doubleHash(char *word);

unsigned long hornerMethod(char *word);

static int calculateDistance (const char * word1,const char * word2);

int main() {
    char *hashTable[15000] = { NULL }; //hash table array
    struct row errorTable[15000] = { NULL }; // error table array
    FILE *fp; // pointer for reading file
    char str[2450];
    char *filename = "D:/Kamran/Lessons/Algoritma Analizi/Assignments/A/smallDictionary.txt"; // get file name
    char * userInput; // tmp variable for storing list values
    int hashValueOfInput; // get hash value of input
    int j=0,i=0; // for while indexes
    int hashValue;
    char arr[5] = " ";
    char *tmp;
    char userInp[350]={NULL};
    //Create list of user input words
    char *listOfUserInput[200]= {NULL};
    char *tmp2;
    int flag = 0;
    int dist; //for levent distance
    char selectedInput[25];
    int selectedHash;


    fp = fopen(filename, "r"); // open file
    // if null print error
    if (fp == NULL) {
        printf("Can not open file %s", filename);
        return 1;
    }
    while (fgets(str, 2450, fp) != NULL);

    fclose(fp);
    //File reading operation has been finished. All string will be stored in str array

    tmp = strtok(str, arr);


    while (tmp != NULL) {
        hashValue = doubleHash(tmp);
        hashTable[hashValue] = tmp;
        tmp = strtok(NULL, arr);
    }


    printf("Please enter text: \n");


    //get user input from command
    gets(userInp);
    // seperate empty spaces from words with strtok
    tmp2= strtok(userInp, arr);
    while (tmp2 != NULL) {
        listOfUserInput[j] = tmp2;
        j++;
        tmp2 = strtok(NULL, arr);
    }

    while(i < j){
        userInput = listOfUserInput[i];
        hashValueOfInput = doubleHash(userInput);
        if(!hashTable[hashValueOfInput])
        {
            //control if the input value is available  in error table or not
            if(errorTable[hashValueOfInput].slct) {
                printf("Selected word: %s",errorTable[hashValueOfInput].slct);

            } else {
                // if it is not available in  table find the nearest word with lev distance
                flag = 0;
                for (i = 0; i< 15000; i++) {
                    if(hashTable[i] != NULL){
                        dist = calculateDistance(userInput, hashTable[i]);
                        if(dist == 1) {
                            flag = 1;
                            printf("%d Levenshtein Edit Distance: %s \n", flag, hashTable[i]);
```

```
                        }
                    }
                }
                if(flag == 0)
                {
                    for (i = 0; i< 15000; i++) {
                        if(hashTable[i] != NULL) {
                            dist = calculateDistance(userInput,hashTable[i]);
                            if(dist == 2) {
                                flag = 2;
                                printf("%d  Levenshtein Edit Distance: %s \n",flag, hashTable[i]);
                            }
                        }
                    }
                }

                if ( flag == 1 || flag == 2)
                {
                    getSelectedWord( userInput ,selectedInput,selectedHash, errorTable,hashTable);
                } else {
                    printf("Can not find word!");
                }
            }
        }
        i++;
    }
    return (0);
}

void getSelectedWord( char * userInput ,char * selectedInput,int selectedHash, struct row *errorTable,char *hashTable){
    printf("Please enter selected word: \n");
    gets(selectedInput);
    selectedHash = doubleHash(selectedInput);
    errorTable[selectedHash].input = userInput;
    errorTable[selectedHash].slct = selectedInput;
    printf("The word %s is converted to  %s \n",errorTable[selectedHash].input, errorTable[selectedHash].slct);
}

int doubleHash(char *word) {
    unsigned long key = hornerMethod(word);
    int h1 = key % 15000;
    int h2 = 1 + (key % (15000 * 15000));
    return (h1 + h2) % 15000;
}

unsigned long hornerMethod(char *word) {
    unsigned long rslt = 1;
    int len = strlen(word);
    int ltrFact;
    int i = 0;
    int j;
    int rFac;
    for (i = 0; i < len - 1; i++) {
        ltrFact = (((word[i]) - 'A') + 1);
        j= 0;
        rFac= 1;
        for (j = (len - 1 - i); j > 0; j--) {
            rFac *= 31;
        }
        rslt += (ltrFact * rFac);
    }
    return rslt;
}

static int calculateDistance (const char * word1,const char * word2)
{
    int userInputLength = strlen(word1);// get length of user input userInputLength
    int hashValueLength = strlen(word2);// get length of hash table value hashValueLength
    int k,i; // loop variables
    int j;
    char c1;
    char c2;
    int min;
    int dlt;
    int insrt;
    int substitute;

    int **mtr =(int**)malloc((userInputLength + 1) * sizeof(int*));
    for ( k = 0; k < userInputLength + 1; ++k) {
        mtr[k]=(int*)malloc((hashValueLength + 1) * sizeof(int));
    }

    for (i = 0; i <= userInputLength; i++) {
        mtr[i][0] = i;
    }
    for (i = 0; i <= hashValueLength; i++) {
        mtr[0][i] = i;
    }
    for (i = 1; i <= userInputLength; i++) {
        c1 = word1[i-1];
        for (j = 1; j <= hashValueLength; j++) {
            c2 = word2[j-1];
```

```
Please enter text:
it is acceptablee
1 Levenshtein Edit Distance: acceptable
Please enter selected word:
acceptable
The word acceptablee is converted to  acceptable

Process finished with exit code 0
```

```
Please enter text:
it is cool
2  Levenshtein Edit Distance: good
2  Levenshtein Edit Distance: code
2  Levenshtein Edit Distance: look
Please enter selected word:
good
The word cool is converted to  good

Process finished with exit code 0
```

```
Please enter text:
it is successfullyyy
2  Levenshtein Edit Distance: successfully
Please enter selected word:
successfully
The word successfullyyy is converted to  successfully

Process finished with exit code 0
```