Report

Data structure

Prepared by:
Ali Atwi

2020-2021

- **Description:**

The following program not only implemented to view the synonyms of a given word but also to add/search... etc. reading from a dynamic file containing some useful english words and their synonyms.

User Manual:

How to use this program?

This program is easy to use:

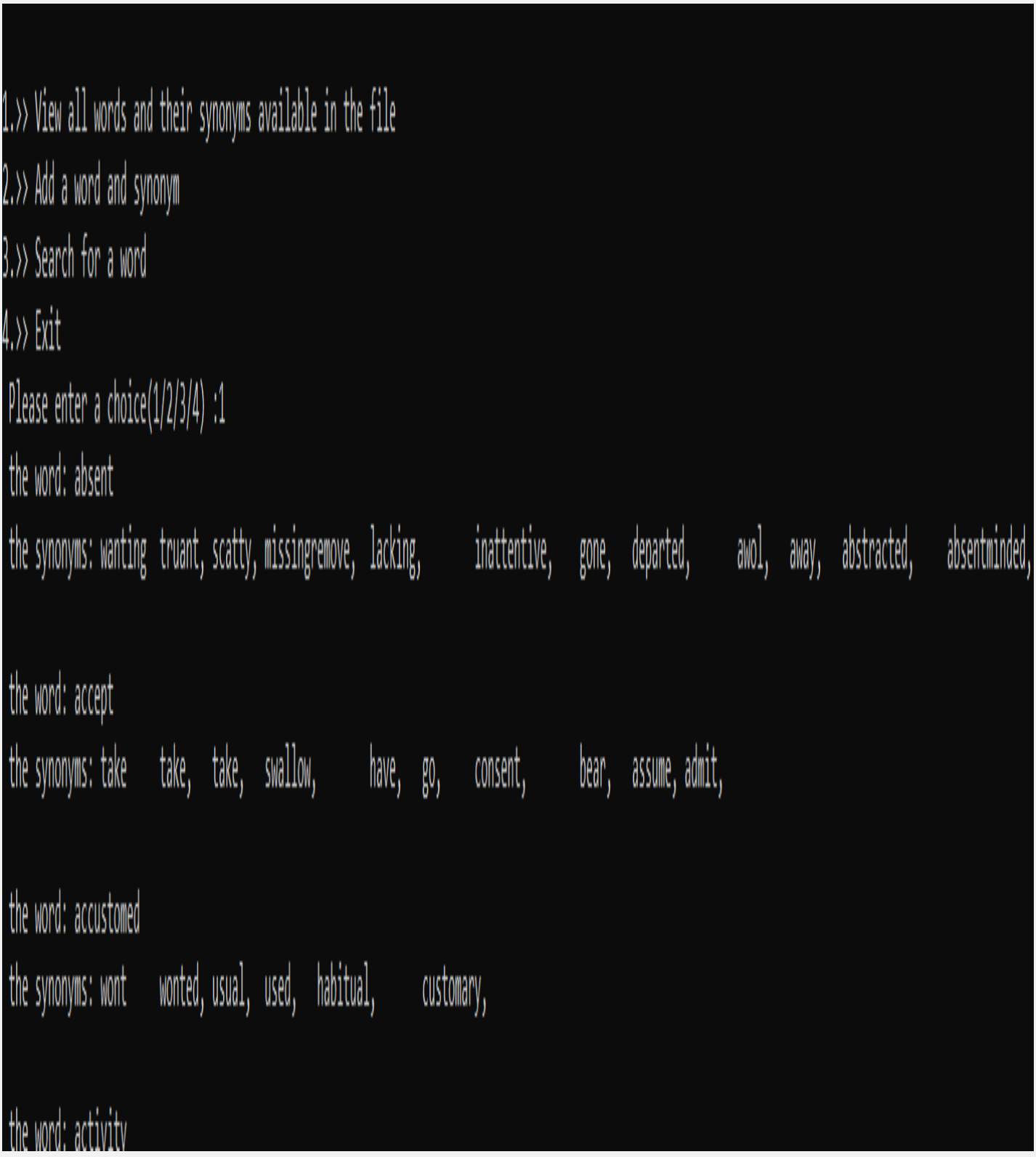
Simply, in order to use this program correctly and without any confusion, and to serve you better; just should follow the given instructions:

First:

In these console screenshots there are list of options...

In order to choose any of them just enter its number then press enter...

>>For displaying the content of the file including the available synonyms please ENTER 1:



>>In case there are missing words you want to add with their synonyms please ENTER 2:

```
1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :2
Enter the word you want to add:Achieve
Enter the synonym/s:Accomplish

1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :3
Enter the word you are searching for:Achieve
synonymes:      Accomplish      (null)
1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :
```

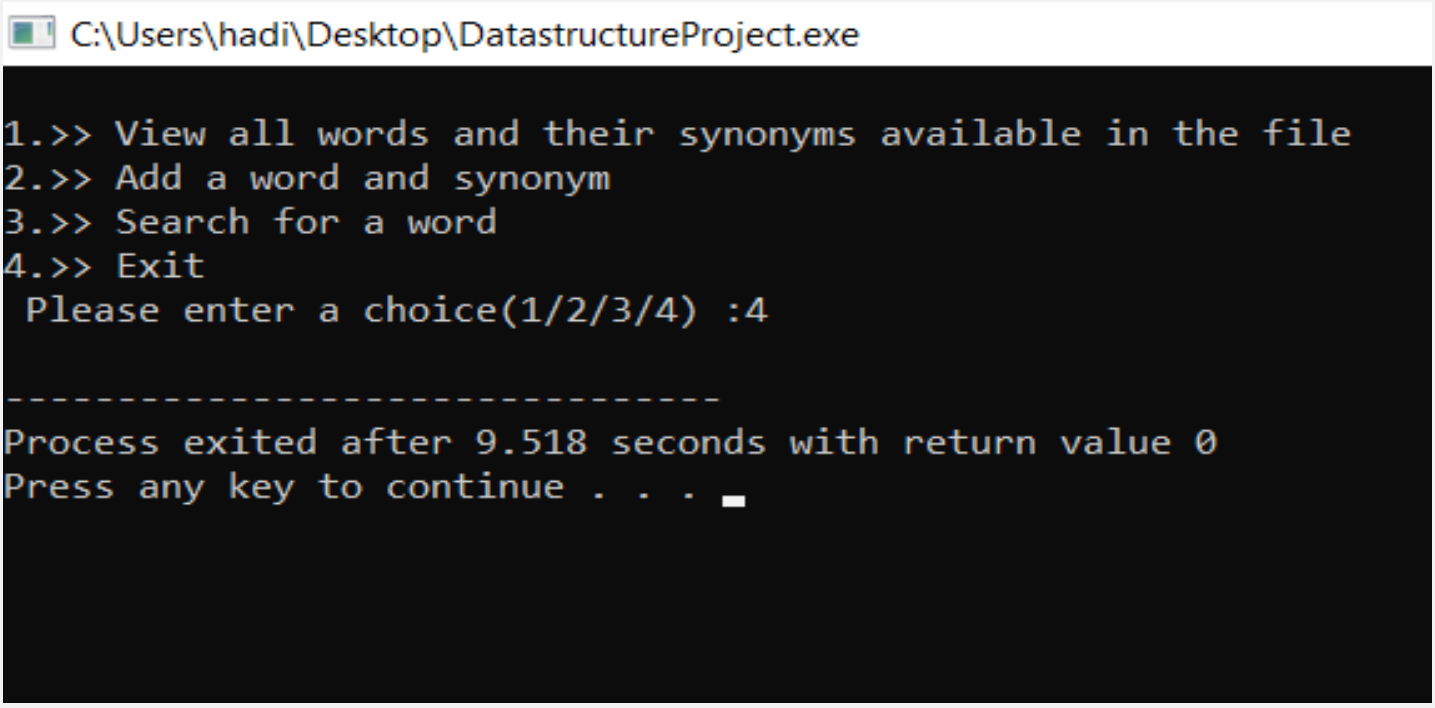
>>If you want to check if a specific word exists in the file or not please ENTER 3:

```
1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :2
Enter the word you want to add:Achieve
Enter the synonym/s:Accomplish

1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :3
Enter the word you are searching for:Achieve
synonymes:      Accomplish      (null)
1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :3
Enter the word you are searching for:try
not found
1.>> View all words and their synonyms available in the file
2.>> Add a word and synonym
3.>> Search for a word
4.>> Exit
 Please enter a choice(1/2/3/4) :
```

Finally:

If you have finished all your work and you want to exit please ENTER 4:



COMPLETED TASKS:

- **Functionality** :
  - ***Main program:***

Main method displays on the console a list of options to be chosen by the user, each option has a specific mission...

In case the user chooses the first choice (show all words and synonyms):

The program will display all the content of the current file which contains every word and its synonym.

The second choice is dedicated if the user wants to add an extra word and it's synonyms in this file.

In case the user have a word and wants to display its synonyms; he should check whether this word in the file .If it is not the case "not found" message will be displayed...

Fourth option is a necessary one in case the user wants to return back...

All these tasks are applied by calling several functions to be discussed briefly

```
int main(int argc, char *argv[]){
int choice;
char*fn="WordsSynonyms.txt";
char word[30];
char synonym[200];

tree a=insertTree(fn);

while (1)
 {
      printf("\n1.>> View all words and their synonyms available in the file\n");
   printf("2.>> Add a word and synonym\n");
   printf("3.>> Search for a word \n");
```

```
printf("4.>> Exit\n Please enter a choice(1/2/3/4) :");
scanf("%d", &choice);

   switch (choice)
   {
      case 1:
      displaySynonyms(a);
      break;
      case 2:
      printf("Enter the word you want to add:");
      scanf("%s",word);
      printf("Enter the synonym/s:");
      scanf("%s",synonym);
      a=Add(a,word,synonym);
      break;
      case 3:
      printf("Enter the word you are searching for:");
      scanf("%s",word);
      Seek(a,word);
      break;
      case 4:exit(0);
```

```
      }
  }
return 0;
}
```

First called function: insertTree(char*name)

This function takes as a parameter name of the file. fscanf reads word by word

if "line" different - so these are the synonyms put them in a created list

```
tree insertTree(char *name)
 {
FILE*fp;
fp=fopen(name,"r");
char line[255];
char word[30];
char synonym[255];
list l;
tree a=NULL;

while(1)
```

```
{
        if(fscanf(fp,"%s",line)==1)
        {
            if(strcmp(line,"-")!=0)
          strcpy(word,line);
       if(strcmp(line,"-")==0)
       {
            l=creatList();
          while(1)
          {
            char t=fgetc(fp);

             if(t!=EOF&&t!='\n')
             {
                  fscanf(fp,"%s",synonym);
                insertStart(&l,synonym);
                          }
             else break;
                  }

          a=Insert(a,word,l);
```

```
        }

    }

  else break;
}

fclose(fp);
return a;
 }
```

The remaining called functions:

- ***Displaying list of synonyms of a word:***

This function has void return type ,it displays all the synonyms of the word.

It works recursively on left and right of a given tree

```
- void displaySynonyms(tree a)
- {
-    printf(" the word: %s \n the synonyms: ",(a));
-    displayList(a->l);
-    printf("\n\n");
-
-        if(!isEmptyTree(Left(a)))
```

```
        displaySynonyms(Left(a));

        if(!isEmptyTree(Right(a)))
        displaySynonyms(Right(a));
    }
```

- ***Displaying lists:***

This function is a helper function for the previous one it's role is displaying a given list.

```
void displayList(list l)
{
    while(l.start!=NULL)
    {
        printf("%s\t",l.start->synonym);
    l.start=l.start->next;
    }

}
```

Insertion:

This function insert a specific word and its synonyms entered by the user.

As we know that RAM is divided into two parts: STACK,HEAP;

So in case of empty tree dynamic allocation takes place in the heap part of the RAM ,this function is able to add a word in the tree nodes and it's synonym in a created list.why list not array ?since we did not know the number of synonyms enter by the user.

```
tree Add(tree a,char word[],char synonym[]){
   if(isEmptyTree(a))
   {
       node1 *n=(node1*)malloc(sizeof(node1));
       strcpy(n->word,word);
       n->left=NULL;
       n->right=NULL;
       list f=creatList();
       insertStart(&f,synonym);
       n->l=f;
       return n;
       }

   else if(strcmp(Root(a),word)==0)
   {
       insertStart(&(a->l),synonym);
       return a;
```

```
    }


  else
  {
      if(strcmp(word,Root(a))<0)
      {
          node1*n=(node1*)malloc(sizeof(node1));
          strcpy(n->word,Root(a));
          n->l=a->l;
          n->right=Right(a);
          n->left=Add(Left(a),word,synonym);
          return n;
          }
      else
      {
          node1*n=(node1*)malloc(sizeof(node1));
          strcpy(n->word,Root(a));
          n->l=a->l;
          n->left=Left(a);
          n->right=Add(Right(a),word,synonym);
          return n;
```

```
            }
        }
}
```

This function is a helper function for the previous one.

```
char insertStart (list *l,char synonym[])
{
      node *n=(node*)malloc(sizeof(node));
   strcpy(n->synonym,synonym);

    if(isEmptyList(*l))
       {
            l->start=l->end=n;
          n->next=NULL;
       }
    else
       {
             n->next=l->start;
          l->start=n;
       }
```

```
    l->size=l->size+1;
}
```

The main major for this function is searching in the tree if a given word exists or not, it works recursively on write and left of the tree in case a word exists in order to display it and all its synonyms

```
int Seek(tree a,char word[])
 {
       if(isEmptyTree(a))
       {
         printf("not found");
      return 0;
       }

    if(strcmp(Root(a),word)==0)
    {
       printf("synonymes:\t");
       printf("%s",returnList(a));
       return 1;
       }
```

```
    else
    {
        if(strcmp(word,Root(a))>0)
        {
            return Seek(Right(a),word);
            }

        else
        {
        return Seek(Left(a),word);
            }

        }

}
```

Part 7 and 6-d are well explained in the program threw comments

Thanks for reading...

best regards