AIWIR WEEK 1

NAME: NAMITHA NAYAK

SRN: PES2UG19CS247

CLASS: D

Word frequency in a given file.

```
f = input("Enter file name with extension: ")
file = open(f, "r")
text = file.read().lower().split()

word = input("Enter the word: ")

def word_count(word, text):
    count = 0
    if word in text:
        for word in text:
            count += 1
    return count

print(word_count(word, text))
```

TEXT FILE:

Hello.txt - Notepad

File Edit Format View Help

```
hello
i am Namitha
hello
PES
HELLO
hello
hello hello hello hello
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\namit\Downloads> python Wordcount.py
Enter file name with extension: Hello.txt
Enter the word: hello
8
PS C:\Users\namit\Downloads>
```

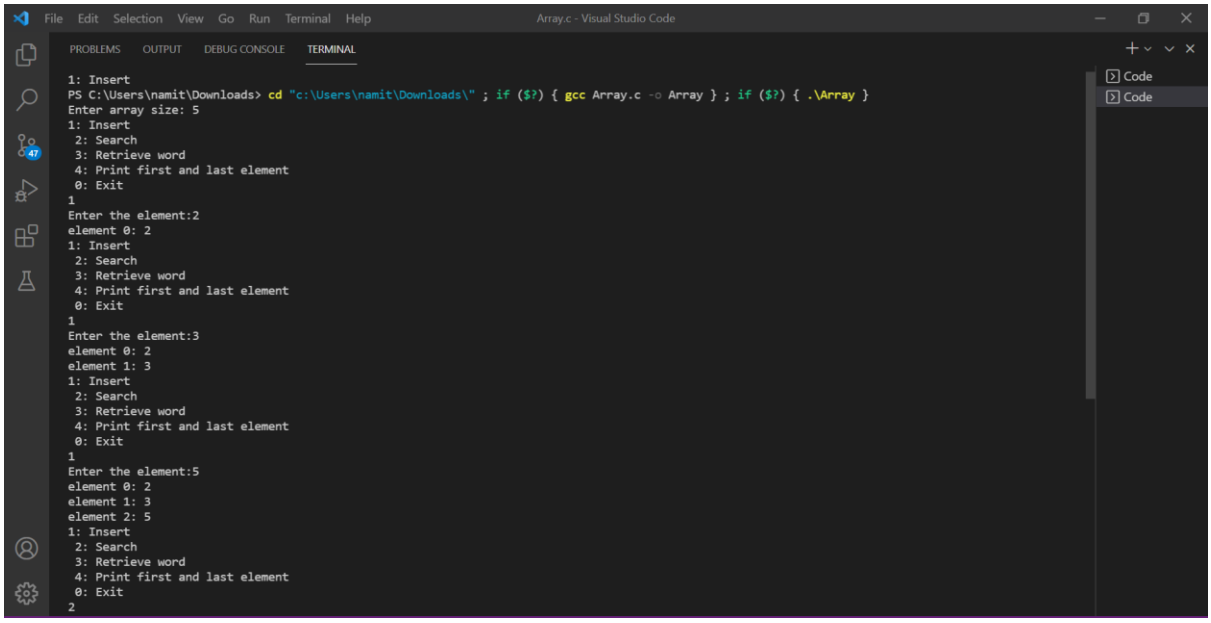
Write a program using array and linked list.

ARRAY:

```c
#include <stdio.h>

#include <stdlib.h>

int insert(int *p, int count, int n){
   printf("Enter the element:");
   scanf("%d", p+count);

   for (int j = 0; j < count+1; j++){
      printf("element %d: %d \n", j, *(p + j));
   }
   return 1;
}

int search(int *p, int count){
   int i;
   printf("Enter element to be searched:");
   scanf("%d", &i);
   for (int j = 0; j < count; j++){
      if (i == *(p+j)){
         printf("Position: %d \n", j);
         return 1;
      }
      else{
         printf("Not found \n");
         return -1;
      }
   }
}

int ind(int *p){
   int i;
   printf("Enter index to be returned:");
   scanf("%d", &i);
   printf("%d \n", *(p+i));
   return 1;
}

int first_last(int* p, int count){
   printf("First element: %d \n", *(p));
   printf("Last element: %d \n", *(p+(count-1)));
}
```

```c
int main(){
   int max, *p, count = 0, c = 0;
   printf("Enter array size: ");
   scanf("%d", &max);
   p = (int*)malloc(max * sizeof(int));


   do{
      printf("1: Insert \n 2: Search \n 3: Retrieve word \n 4: Print first and
last element \n 0: Exit \n");
      scanf("%d", &c);
      switch (c){
      case 1:
         insert(p, count, max);
         count++;
         break;
      case 2:
         search(p, count);
         break;
      case 3:
         ind(p);
         break;
      case 4:
         first_last(p, count);
         break;
      }
   if (c == 0){
      free(p);
      return 0;
   }
   }while(c != 0);

}
```

OUTPUT:



LINKED LIST:

Main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include"sll.h"

int main() {
    int choice;
    int count = 0;
    List* list = list_initialize();
    do {
        printf("1: Insert \n 2: Print List \n 3: Search \n 4: Retrieve word \n
5: Print first and last element \n 0: Exit \n");
        scanf("%d", &choice);
        switch(choice) {
            int element, index;
            case 1:
                /*Insert element at the End of the list*/
```

```
                printf("Enter element: ");
                scanf("%d", &element);
                count++;
                insert_at_end(list, element);
                break;
            case 2:
                /* Print list contents */
                list_print(list);
                break;
            case 3:
                printf("Enter element to be searched:");
                scanf("%d", &element);
                printf("%d \n", search_list(list, element));
                break;
            case 4:
                printf("Enter index to be retrieved: ");
                scanf("%d", &index);
                printf("The element is: %d \n", retrieve_element(list,
index));
                break;
            case 5:
                printf("The first element is: %d \n", retrieve_element(list,
0));
                printf("The last element is: %d \n", retrieve_element(list,
count - 1));
                break;
        }
    } while(choice != 0);
    list_destroy(list);
    return 0;
}

List* list_initialize() {
    List* list = (List*) malloc(sizeof(List));
    list->head = NULL;
    list->number_of_nodes = 0;
    return list;
}

void list_print(List* list)
    {
    Node *p;
    p=list->head;
    if(p == NULL)
    {
        printf("EMPTY\n");
        return;
    }
```

```
    while (p!=NULL){
        printf("%d ",p->data);
        p=p->link;
    }
    printf("\n");
}


void list_destroy (List *list)
{
    Node *t, *u=NULL;
    t=list->head;
    while (t->link!=NULL){
        u=t;
        t=t->link;
        free(u);
    }
    free(list);
}
```

Sll.c:

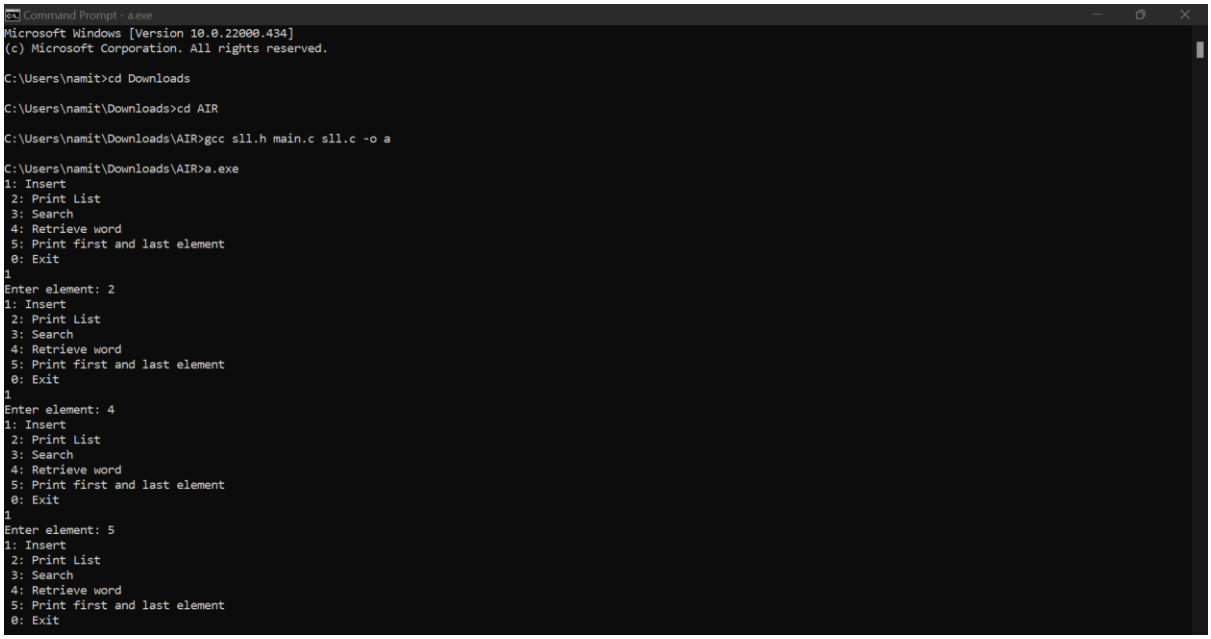
```
#include<stdio.h>
#include<stdlib.h>
#include "sll.h"

void insert_at_end(List *list, int data) {
    Node* newNode;
    newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL){
        printf("Unable to allocate memory\n");
    }
    else{
        Node* temp = NULL;
        temp = list->head;
        newNode->data=data;
        newNode->link=NULL;
        if(temp == NULL){
            list->head = newNode;
        }
        else{
            while(temp!=NULL && temp->link!=NULL)
                temp = temp->link;
            temp->link = newNode;
        }
        list->number_of_nodes += 1;
    }
}
```

```c
int search_list(List* list, int element){
   Node* temp = NULL;
   temp = list->head;
   int count = 0;
   while(temp != NULL){
       if (temp->data == element){
           return count;
       }
       temp = temp->link;
       count++;
   }
   return -1;
}

int retrieve_element(List* list, int index){
    Node* temp = NULL;
    temp = list->head;
    while(index != 0 && temp != NULL){
        temp = temp->link;
        index --;
    }
    return temp->data;
}
```

Sll.h:

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

typedef struct Node {
    int data;
    struct Node *link;
} Node;

typedef struct List {
    Node *head;
    int number_of_nodes;
} List;

List* list_initialize();
void insert_at_end(List *list, int data);
void list_print(List* list);
void list_destroy(List* list);
int search_list(List* list, int element);
int retrieve_element(List* list, int index);
```

OUTPUT:


Command Prompt - a.exe
Microsoft Windows [Version 10.0.22000.434]
(c) Microsoft Corporation. All rights reserved.
C:\Users\namit>cd Downloads
C:\Users\namit\Downloads>cd AIR
C:\Users\namit\Downloads\AIR>gcc sll.h main.c sll.c -o a
C:\Users\namit\Downloads\AIR>a.exe
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit
1
Enter element: 2
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit
1
Enter element: 4
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit
1
Enter element: 5
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit


Command Prompt - a.exe
0: Exit
2
2 4 5
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit
3
Enter element to be searched:4
1
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit
4
Enter index to be retrieved: 0
The element is: 2
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit
5
The first element is: 2
The last element is: 5
1: Insert
2: Print List
3: Search
4: Retrieve word
5: Print first and last element
0: Exit