

**Department of CSE**

Subject: Data Structures and Algorithms

Assignment 3

Semester: 3rd Sem

Session: August-December 2018

Submitted by : Rahul bera

Roll no. : 17CSE52

Section : A1

Submitted to : Ms. Ekta

1. Write a program in C language that can do the following tasks:

* reads a file
* breaks each line into words
* strips whitespace and punctuation from the words
* Converts them to lowercase.

Input: The File is attached below.

Output: As mentioned above.

**Code:**

#include <stdio.h> // for i/o

#include <ctype.h> // for string processing

void readbyword(FILE \*fp);

void removepunct(FILE \*fp);

char lower(char c);

int main(){

FILE \*fp; // file pointer

int choice;

fp = fopen("file.txt","r"); // opens the file.txt file , "r" specifies to open in read mode

printf(" \n 1. Print file word by Word\n 2. Strip whitespace and punctuation & print in lowercase\n\n Enter your choice : ") ;

scanf("%d",&choice);

switch(choice){

case 1: readbyword(fp);

break;

case 2: removepunct(fp);

break;

default : printf("enter correct !! bye");

}

fclose(fp); // closing of file

return 0;

}

void readbyword(FILE \*fp){ // function to read file and output each word

char c;

while((c = fgetc(fp)) != EOF){

if(c == ' ' || c == '\n')

{

printf("\n");

}

else

{

printf("%c", c);

}

}

}

void removepunct(FILE \*fp){

char str;

while((str = fgetc(fp)) != EOF){

// this line is checking if str is not a punctuation or whitespace character

if(!ispunct(str) && str!=' ' && str!='\n' && str!='\t' && str!='\v' && str!='\f' && str!='\r'){

str = lower(str);

printf("%c",str);

}

else{

// if whitespace or punctuation in encountered it prints newline

printf("\n");

}

}

}

char lower(char c){ // function to return the lowercase of passed argument

if(islower(c)){

return c;

}

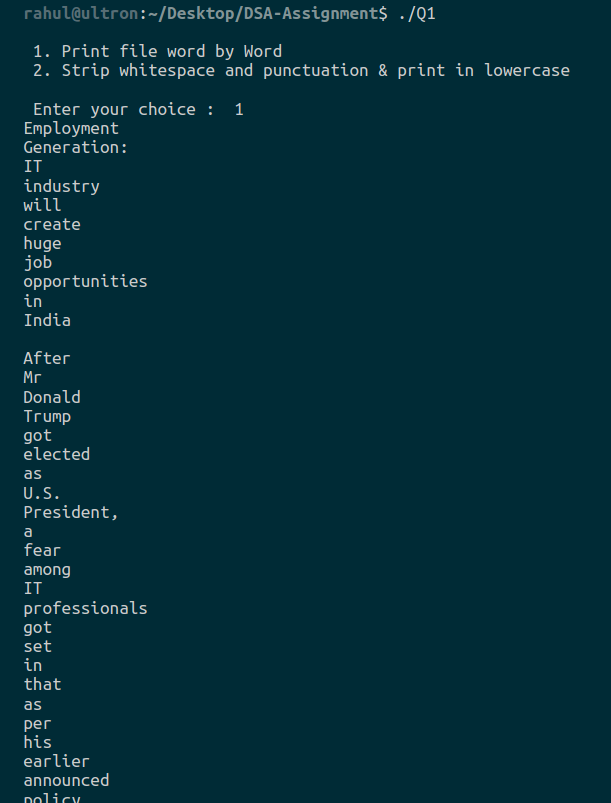
else{

return tolower(c);

}

}

output :





2. There is an array given with the following elements: 20, 15, 25, 2, 30, 95, 41, 68, 4, 39. The array is an unsorted array. Write a program in C which searches the element 41 and 65 from the array with the total time complexity of O(n).

**Code:**

/\*

Find given numbers in a given array in O(n) complexity.

\*/

#include <stdio.h>

int main(){

int flag1,flag2,a[]= {20, 15, 25, 2, 30, 95, 41, 68, 4, 39};

flag1=flag2=0; // counter variables

for(int i=0; i<10; i++){ // only 1 for loop is used therefore complexity : O(n)

if(a[i] == 41){

printf("41 Found at %dth position\n",i+1);

flag1 = flag1 +1;

}

if(a[i] == 65){

printf("65 found at %dth position\n",i+1);

flag2 = flag2 +1;

}

}

if(flag1 == 0){ // counter for 41

printf("41 not found\n");

}

if(flag2 == 0){ // counter for 62

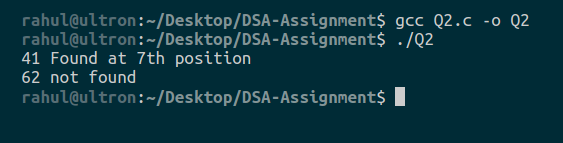
printf("62 not found\n");

}

return 0;

}

output :



3. There is an array of 12 elements. The elements are stored from A[2] to A[14] positions or locations in the memory. We need to delete an element from the array at the location A[7]. Write a program which delete the element at the location given and the deletion process must reduce the size of the array so that this chunk of memory can be used by other elements.

**Code:**

#include<stdio.h>

#include<stdlib.h>

void main()

{

//Initialize the array

int \*array;

int i,j;

int size;

int delete;

printf("Enter the size of array\n");

scanf("%d", &size);

array = (int \*)calloc(size,sizeof(int));

printf("Enter elements of array\n");

for(i=0; i<size; i++)

scanf("%d", &array[i]);

printf("Elements are: ");

for(i=0; i<size; i++)

printf("%d ", array[i]);

printf("\n");

delete = array[3];

for(i=0; i<size; i++){

if(array[i] == delete){

for(j=i; j<size; j++){

array[j] = array[j+1];

}

}

}

//array = realloc(array, (size-1)\*sizeof(int));

printf("\nAfter deletion the array is:");

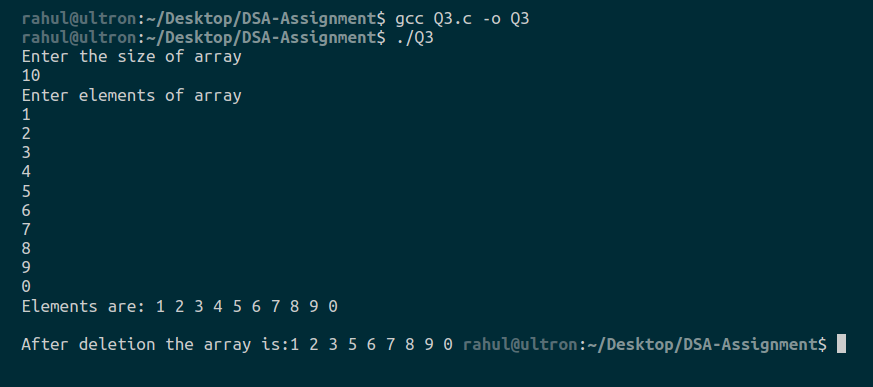
for(i=0; i<size-1; i++)

printf("%d ", array[i]);

free(array);

}

output :



4. Consider a link list with 6 elements. After inserting all the elements in the list, the elements are now taken out of the list one by one. The elements when all deleted are arranged in a way that the one which is inserted at first iteration comes at first spot and so on. Apply this first in first out strategy and display the result. The operations must be done in the following order:

* Insert()- insert the 6 elements in the list
* Delete()- delete elements one by one
* Find out the method in data structure which can use this delete() operation and apply a first in first out strategy
* Output will be an empty list.

**Code:**

/\*

I have to implement a data structure that allows FiFo (First In First Out) stratergy.

Therefore i implemented Queue ;)

\*/

#include <stdio.h> // for i/o operations

#include <malloc.h> // for memory allocation

struct Queue

{

int data;

struct Queue \*next;

};

typedef struct Queue ds;

ds \*head = NULL;

ds \*last = NULL;

void insert(int n);

void delete();

void display();

// Driver code

int main(){

insert(1);

insert(2);

insert(3);

insert(4);

insert(5);

insert(6);

printf(" linked list is : \n");

display();

printf("\n");

delete();

display();

delete();

display();

delete();

display();

delete();

display();

delete();

display();

delete();

display();

return 0;

}

void insert(int n){

ds \*newnode = (ds \*)malloc(sizeof(ds));

newnode->data = n;

if(head == NULL){

head = last = newnode;

newnode->next = NULL;

}

else{

last->next = newnode;

newnode->next = NULL;

last = newnode;

}

}

void delete(){

ds \*ptr = head;

head = head->next;

printf("\n deleting : %d ",ptr->data);

printf("\n");

free(ptr);

}

void display(){

if(head == NULL){

printf("Nothing to show \n");

}

else{

ds \*ptr = head;

// traversing the list till null is encountered

while(ptr->next != NULL){

printf("%d -> ",ptr->data);

ptr = ptr->next;

}

printf(" %d",ptr->data);

}

}

output :

