

**BHARATI VIDYAPEETH’S**

**INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT**

(Affiliated to Guru Gobind Singh Indraprastha University, Approved by AICTE, New Delhi)

**Data and File Structures**

**(MCA-162)**

**Practical File**

**Submitted To: Submitted By:**

Dr. Sunit Pratap Singh Deepak

(Enrolment No. -04335304421

(Associate Professor) MCA 2nd SEM SEC II

| S.No | Description | Date | Remark |
| --- | --- | --- | --- |
| [AP1](#bookmark=id.iqt40m5zkngw) | Write a program which takes an array of n integers and displays the frequency of each element present in the array. |  |  |
| [AP2](#bookmark=id.o32b6xqpn3pa) | Write a program which takes an array of n integers and performs searching of an element by implementing linear search and binary search techniques |  |  |
| [AP3](#bookmark=id.spda3ky2sm7h) | Write a program which takes an array of n integers and sorts the integers in descending order using bubble sort and selection sort techniques. |  |  |
| [AP4](#bookmark=id.1j2cexeo4g2n) | Write a program which takes an array of n integers and sorts the integers in ascending order using insertion sort technique. |  |  |
| [AP5](#bookmark=id.c659f43s3zp9) | Write a program which takes an array of n integers and sorts the integers in ascending order using quick sort technique |  |  |
| [BP1](#bookmark=id.amdhbzxsfk95) | Write a menu-driven program which implements a linear linked list with following operations: a) Insertion of an element at beginning of the list b) Insertion of an element at specific location of the list c) Insertion of an element at end of the list d) Deletion of an element from the beginning of the list e) Deletion of an element from specific location of the list f) Deletion of an element from the end of the list g) Display all elements of the list h) Search a specific element in the list |  |  |
| [BP2](#bookmark=id.kwe5wwd4qph1) | A polynomial is composed of different terms where each of them holds a coefficient and an exponent. Write a program to represent the following polynomials: 4x4 + 4x3 -2x2 + x and 11x3 + 7x2 - 4x with linear linked list, and then perform addition of the given polynomials. |  |  |
| [CP1](#bookmark=id.gtnlzkhqmpzl) | Write a menu-driven program which implements a stack (using onedimensional array) with following operations: a) Push (insert an element) b) Pop (delete an element) c) Display (print all the elements of stack) |  |  |
| [CP2](#bookmark=id.rv7c6u7dt7cp) | Write a menu-driven program which implements a linear queue (using one-dimensional array) with following operations: a) Enqueue (insert an element) b) Dequeue (delete an element) c) Display (print all the elements of queue) |  |  |
| [CP3](#bookmark=id.6a1lnsl2dlkc) | Write a menu-driven program which implements a circular queue (using one-dimensional array) with following operations: a) Enqueue (insert an element) b) Dequeue (delete an element) c) Display (print all the elements of queue) |  |  |

AP1.

#include <stdio.h>

int main()

{

int arr[10], freq[10];

int i, j, count;

printf("this program is coded by Deepak(04335304421)\n ");

// Input 10 of array

// Input elements in array

printf("Enter elements in array 10:\n ");

for(i=0; i<10; i++)

{

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

/\* Initially initialize frequencies to -1 \*/

freq[i] = -1;

}

for(i=0; i<10; i++)

{

count = 1;

for(j=i+1; j<10; j++)

{

// If duplicate element is found

if(arr[i]==arr[j])

{

count++;

// Make sure not to count frequency of same element again

freq[j] = 0;

}

}

/\* If frequency of current element is not counted \*/

if(freq[i]!= 0)

{

freq[i] = count;

}

}

// Print frequency of each element

printf("\nFrequency of all elements of array : \n");

for(i=0; i<10; i++)

{

if(freq[i] != 0)

{

printf("%d occurs %d times\n", arr[i], freq[i]);

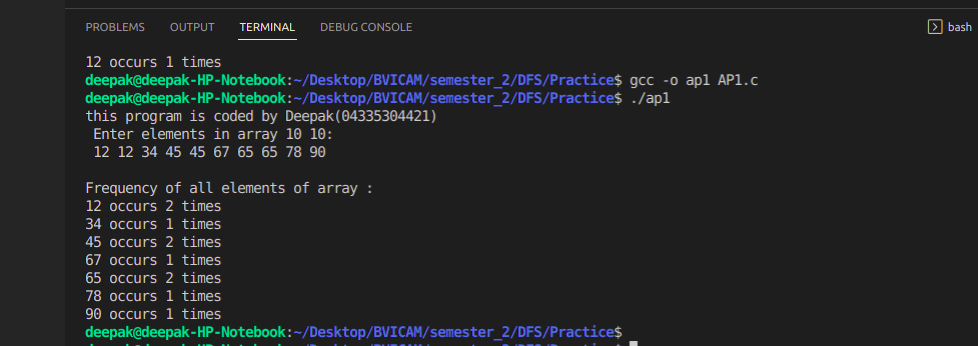
}

}

return 0;

}

Output:-



AP2

#include<stdio.h>

void linearSearch(int arr[],int item){

int i, j,size;

size= 10;

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

if(arr[i]==item){

printf("Item Found\n");

return;

}

}

printf("Item Not Found.\n");

}

void binarySearch(int arr[], int item){

int left, right, mid,i,j,size,a;

size= 10;

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

for (j = i + 1; j < size; j++){

if (arr[i] > arr[j]){

a = arr[i];

arr[i] = arr[j];

arr[j] = a;

}

}

}

left=0;

right=size-1;

while(left<=right){

mid=(left+right)/2;

if(arr[mid]==item){

printf("Item Found");

return;

}

else if(arr[mid]<item){

left=mid+1;

}

else

right=mid-1;

}

printf("item Not Found");

}

void main(){

int arr[10],i, j,item,choice;

printf("Coded by Deepak(04335304421)\n");

//Enter the element of array;

printf("Enter 10 Elements of array:\n");

for(i=0;i<10;i++){

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

}

//searching choice

printf("enter element to Search in array");

scanf("%d",&item);

do{

printf("Searching type:-\n");

printf("1. linear Search.\n2. for Binary Search\n3. exit");

scanf("%d",&choice);

switch (choice)

{

case 1:

linearSearch(arr,item);

break;

case 2:

binarySearch(arr,item);

break;

default:

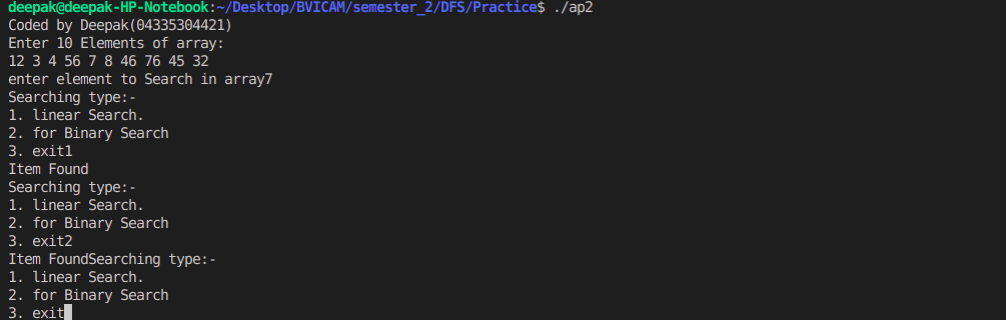
choice=3;

}

}while(choice!=3);

}

Output:-



AP3.

#include<stdio.h>

//bubble sort

void displayArr(int arr[], int);

void bubbleSort(int arr[], int num){

int x, y, temp;

for(x = 0; x < num - 1; x++){

for(y = 0; y < num - x - 1; y++){

if(arr[y]<arr[y + 1]){

temp = arr[y];

arr[y] = arr[y + 1];

arr[y + 1] = temp;

}

}

}

displayArr(arr,num);

}

//selection sort

void selectionSort(int arr[],int num){

int i, j,temp;

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

for (j = i+1; j < num; j++)

{

if (arr[i]<arr[j])

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

displayArr(arr,num);

}

//display array

void displayArr(int arr[],int num){

int i;

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

{

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

}

}

void main(){

int i, j, arr[10],item,choice;

printf("Coded by Deepak(04335304421)\n");

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

//enter array element

for(i=0; i<10;i++){

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

}

// select choice

do{

printf("Sorting type:-\n");

printf("1. Bubble Sort.\n2.Selection Sort \n3. exit");

scanf("%d",&choice);

switch (choice)

{

case 1:

bubbleSort(arr,10);

break;

case 2:

selectionSort(arr,10);

break;

default:

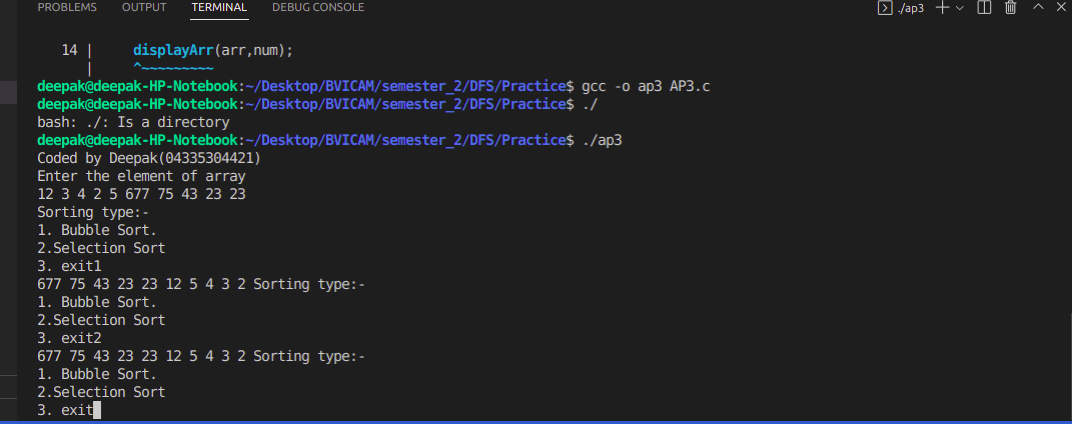
choice=3;

}

}while(choice!=3);

}

Output:-



AP4

#include<stdio.h>

void displayArr(int arr[], int);

//insertion sort function

void insertionSort(int arr[],int num){

int i,temp,j;

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

{

temp=arr[i];

j=i-1;

while (j>=0 && arr[j]>temp)

{

arr[j+1]=arr[j];

j--;

}

arr[j+1]=temp;

}

displayArr(arr,num);

}

//display element

void displayArr(int arr[],int num){

int i;

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

{

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

}

}

void main(){

int i, j,arr[10], num=10;

printf("Code by Deepak(04335304421)\n");

printf("Enter the Elements of Array");

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

{

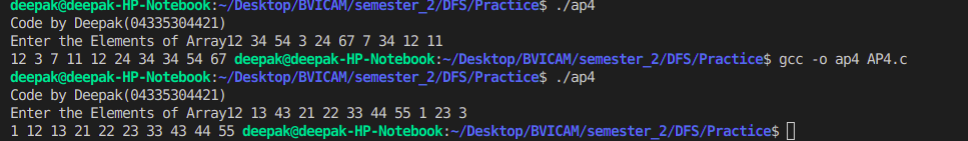
scanf("%d",&arr[i]);

}

insertionSort(arr,num);

}

Output:-



AP5.

#include <stdio.h>

void swap(int \*a, int \*b) {

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition(int array[], int low, int high) {

int j;

int pivot = array[high];

int i = (low - 1);

for ( j = low; j < high; j++) {

if (array[j] <= pivot) {

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

void quickSort(int array[], int low, int high) {

if (low < high) {

int pi = partition(array, low, high);

quickSort(array, low, pi - 1);

quickSort(array, pi + 1, high);

}

}

void printArray(int array[], int size) {

int i;

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

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

}

printf("\n");

}

// main function

int main() {

int data[] = {8, 7, 2, 1, 0, 9, 6};

int n = sizeof(data) / sizeof(data[0]);

printf("Code by Deepak(04335304421)");

printf("Unsorted Array\n");

printArray(data, n);

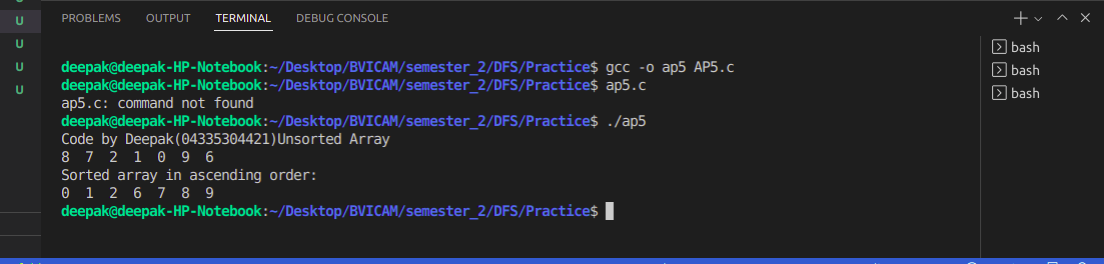
quickSort(data, 0, n - 1);

printf("Sorted array in ascending order: \n");

printArray(data, n);

}

output:-



BP1.

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

struct node

{

int data;

struct node \*next;

};

typedef struct node NODE;

NODE \*start = NULL;

void insBegining(int item)

{

NODE \*node;

node = (NODE\*)malloc(sizeof(NODE));

node->data = item;

if(start==NULL)

{

node->next = NULL;

}

else

{

node->next = start;

}

start = node;

printf("\nNODE INSERTED");

}

void Lastinsert(int item)

{

NODE \*ptr=(NODE\*)malloc(sizeof(NODE));

NODE\*temp;

if(ptr == NULL )

{

printf("\n OVERFLOW");

}

else

{

ptr->data=item;

if(start == NULL)

{

ptr->next=NULL;

start=ptr;

printf("\nNODE INSERTED");

}

else

{

temp = start;

while(temp->next != NULL)

{

temp = temp->next;

}

temp->next=ptr;

ptr->next = NULL;

printf("NODE INSERTED");

}

}

}

void displayList()

{

NODE \*temp;

temp = start;

if(temp == NULL)

{

printf("\n LIST IS EMPTY!");

}

else

{

printf("\n LIST: ");

while(temp != NULL)

{

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

temp = temp -> next;

}

printf("NULL");

}

}

void delbeg()

{

NODE \*ptr;

if(start == NULL)

{

printf("\n List is Empty");

}

else

{

ptr=start;

start= start -> next;

free(ptr);

printf("\n Node deleted from the begining...");

}

}

void enddel()

{

NODE \*ptr,\*ptr1;

if(start == NULL)

{

printf("\nList is empty");

}

else if(start -> next ==NULL)

{

start=NULL;

free(start);

printf("\nOnly node of the list deleted");

}

else

{

ptr=start;

while(ptr->next != NULL)

{

ptr1=ptr;

ptr=ptr->next;

}

ptr1->next= NULL;

free(ptr);

printf("\n Deleted node from the last");

}

}

void specificins(int item, int loc)

{

NODE \*ptr=(NODE \*)malloc(sizeof(NODE));

NODE \*temp;

int i;

if(ptr == NULL)

{

printf("\n Overflow");

}

else

{

ptr->data=item;

temp=start;

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

{

temp=temp->next;

if(temp==NULL)

{

printf("\nCan't insert");

return;

}

}

ptr->next=temp->next;

temp->next=ptr;

printf("\n Node inserted");

}

}

void specificdel(int loc)

{

NODE \*ptr,\*ptr1;

int i;

ptr=start;

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

{

ptr1=ptr;

ptr=ptr->next;

if(ptr==NULL)

{

printf("\nThere are less than %d elements in the list",loc);

return ;

}

}

ptr1->next=ptr->next;

free(ptr);

printf("\nNode Deleted");

}

void search()

{

NODE \*ptr;

int item,i=0,flag=1;

ptr=start;

if(ptr==NULL)

{

printf("\nEmpty list");

}

else

{

printf("\nEnter item which you want to search?");

scanf("%d",&item);

while(ptr!=NULL)

{

if(ptr->data==item)

{

printf("\n Item found at location %d ",(i));

flag=0;

}

i++;

ptr=ptr->next;

}

if(flag!=0)

{

printf("\nItem not found");

}

}

}

void main()

{

int choice,loc,item;

printf("Coded By Deepak(04335304421)\n");

do

{

printf("\n\n\*\*\*\*\*\*\*\*\*\*OPTIONS\*\*\*\*\*\*\*\*\*\*");

printf("\n1. INSERT AT BEGINING");

printf("\n2. INSERT AT SPECIFIC LOCATION");

printf("\n3. INSERT AT END");

printf("\n4. DELETE FROM BEGINING");

printf("\n5. DELETE FROM SPECIFIC LOCATION");

printf("\n6. DELTE FROM END");

printf("\n7. SEARCH THE ELEMENT");

printf("\n8. DISPLAY THE LIST");

printf("\n9.Exit");

printf("\n ENTER YOUR CHOICE ");

scanf("%d", &choice);

switch(choice)

{

case 1:printf("\n Input the Node Data: ");

scanf("%d",&item);

insBegining(item);

break;

case 2:printf("\n Input the Node Data: ");

scanf("%d",&item);

printf("\n Enter the location:");

scanf("%d",&loc);

specificins(item,loc);

break;

case 3:printf("\n Input the Node Data: ");

scanf("%d",&item);

Lastinsert(item);

break;

case 4:delbeg();

break;

case 5:

printf("\nEnter the location:");

scanf("%d",&loc);

specificdel(loc);

break;

case 6:enddel();

break;

case 7:search();

break;

case 8:

displayList();

break;

case 9:exit(0);

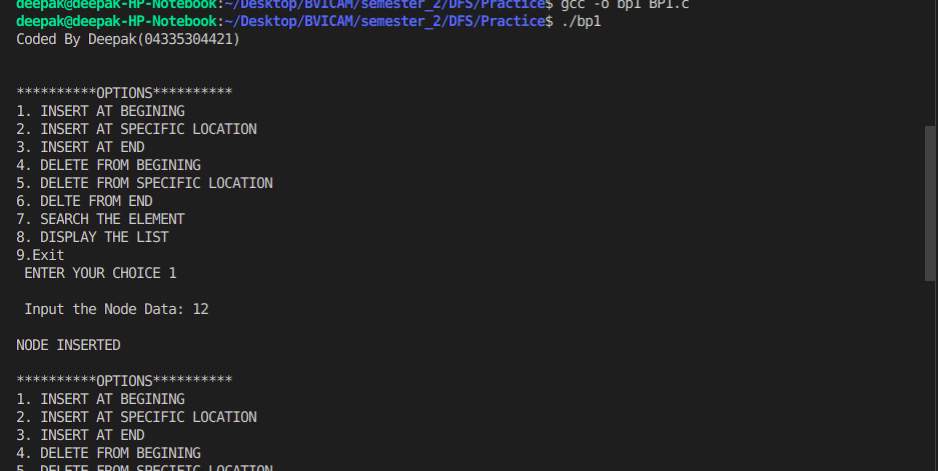
default : printf("\n WRONG CHOICE");

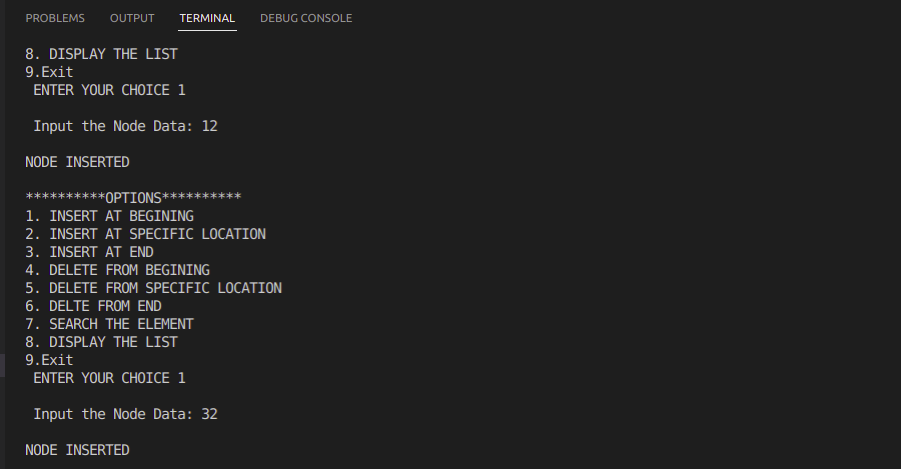
}

}while(choice!=10);

}

output:-





BP2.

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct Node

{

int coff;

int pow;

struct Node \*next;

};

typedef struct Node NODE;

void createPoly(NODE \*\*);

void printPoly(NODE \*);

void addPoly(NODE \*\*, NODE \*, NODE \*);

void main()

{

printf("Coded by Deepak(04335304421)\n");

NODE \*p, \*q, \*r;

printf("\n CREATE FIRST POLYNOMIAL");

createPoly(&p);

printf("\n FIRST POLYNOMIAL");

printPoly(p);

printf("\n CREATE SECOND POLYNOMIAL");

createPoly(&r);

printf("\n SECOND POLYNOMIAL");

printPoly(r);

printf("\n ADDITION OF THE POLYNOMIAL");

addPoly(&q,p,r);

printPoly(q);

}

void createPoly(NODE \*\*start)

{

int flag;

int coe, pow;

NODE \*node=(NODE \*)malloc(sizeof(NODE));

\*start=node;

do{

printf("\n INPUT THE COFFICIENT: ");

scanf("%d",&coe);

printf("\n INPUT THE POWER: ");

scanf("%d",&pow);

node->coff=coe;

node->pow=pow;

node->next=NULL;

//clrscr();

printf("\n DO YOU WANT TO ADD MORE TERMS TO THE POLYNOMIAL(0/1)");

scanf("%d",&flag);

if(flag)

{

NODE \*newNode = (NODE \*)malloc(sizeof(NODE));

node->next = newNode;

node = newNode;

node->next = NULL;

}

}while(flag);

}

void printPoly(NODE \*node)

{

printf("\n POLYNOMIAL EXPRESSION: ");

while(node!=NULL)

{

printf("%dx^ %d",node->coff,node->pow);

node=node->next;

if(node!=NULL)

{

printf(" + ");

}

}

}

void addPoly(NODE \*\*start, NODE \*p, NODE \*q)

{

NODE \*node=(NODE \*)malloc(sizeof(NODE));

node->next=NULL;

\*start=node;

while(p && q)

{

if(p->pow > q->pow)

{

node->pow = p->pow;

node->coff = p->coff;

p=p->next;

}

else if(p->pow < q->pow)

{

node->pow = q->pow;

node->coff = q->coff;

q=q->next;

}

else{

node->pow=p->pow;

node->coff=p->coff + q->coff;

p=p->next;

q=q->next;

}

if(p && q)

{

NODE \*newNode = (NODE \*)malloc(sizeof(NODE));

node->next = newNode;

node = newNode;

node->next = NULL;

}

}

while(p || q)

{

NODE \*newNode = (NODE \*)malloc(sizeof(NODE));

node->next = newNode;

node = newNode;

node->next = NULL;

if(p)

{

node->pow=p->pow;

node->coff=p->coff;

p=p->next;

}

if(q)

{

node->pow=q->pow;

node->coff=q->coff;

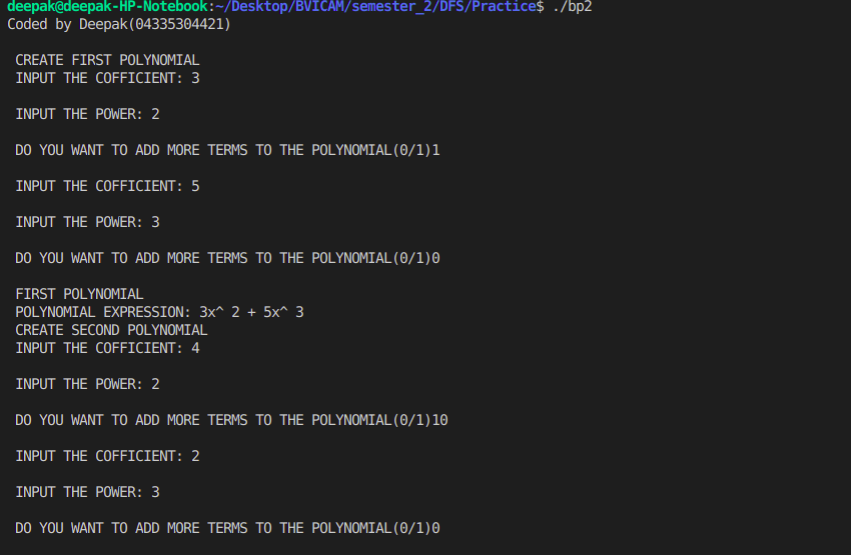
q=q->next;

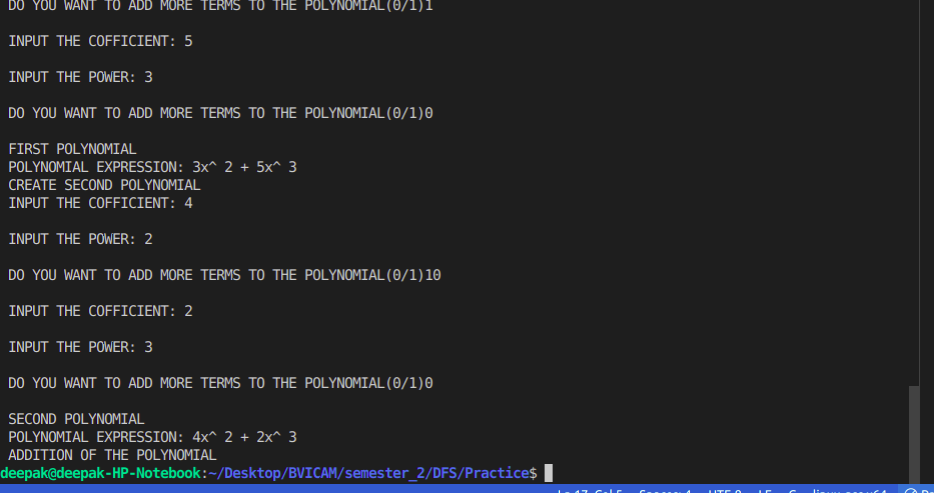
}

}

}

output:-





CP1.

//stack implementation using array

#include<stdio.h>

int a[10], size=10,top=-1,i;

int pop();

void push(int);

int isFull();

int isEmpty();

void display();

void main(){

int choice, item;

printf("coded by Deepak(04335304421)");

do{

printf("\n\*\*\*\*\*\*\*\*\*\*\*Menu\*\*\*\*\*\*\*\*\*\*");

printf("\n1. Push");

printf("\n2. Pop");

printf("\n3. Display Stack element");

printf("\n4. exit");

printf("\nEnter your Choice");

scanf("%d",&choice);

switch(choice){

case 1:

printf("\nEnter element to Push");

scanf("%d",&item);

push(item);

display();

break;

case 2:

printf("\npop element is %d",pop());

display();

break;

case 3:

printf("\nElements of stack is ");

display();

break;

default:

choice =4;

}

}while(choice!=4);

}

void push(int ele){

if(isFull()){

printf("\nstack overflow");

}

top++;

a[top]=ele;

}

int pop(){

int temp;

if(isEmpty()){

printf("\nunderflow");

// return ;

}else{

temp=a[top];

top--;

return temp;

}

}

int isEmpty(){

if(top==-1){

return 1;

}

return 0;

}

int isFull(){

if(top==(size)-1){

return 1;

}

return 0;

}

void display(){

if(isEmpty()){

printf("\nStack is empty");

}

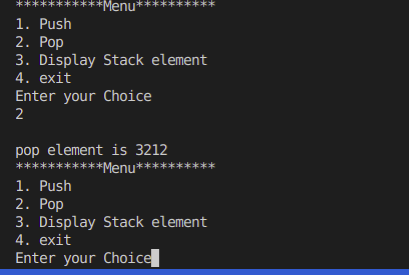
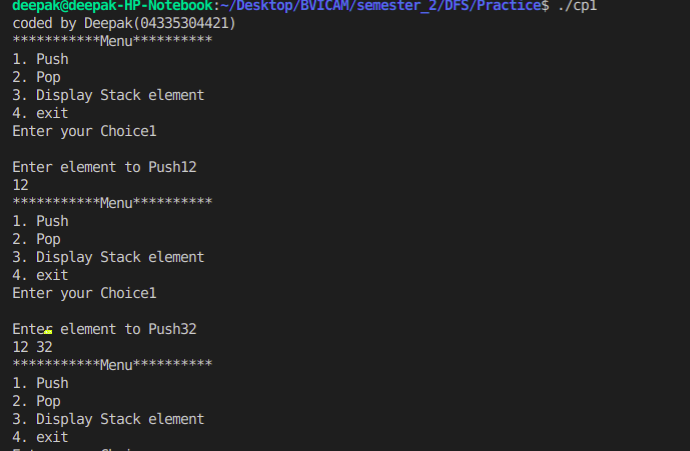
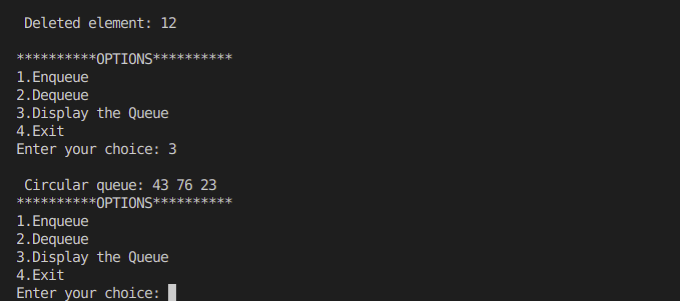
for(i=0;i<=top;i++){

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

}

}

Output:-



CP2.

//stack implementation using array

#include<stdio.h>

// #include<conio.h>

int a[10], size=10,rear=-1,front=-1,i;

int dequeue();

void enqueue(int);

// void peep();

int isFull();

int isEmpty();

void display();

void main(){

int choice, item;

printf("coded by Deepak(04335304421)");

do{

printf("\n\*\*\*\*\*\*\*\*\*\*\*Menu\*\*\*\*\*\*\*\*\*\*");

printf("\n1. Enqueue");

printf("\n2. Dequeue");

printf("\n3. Display Queue element");

printf("\n4. exit");

printf("\nEnter your Choice");

scanf("%d",&choice);

switch(choice){

case 1:

printf("\nEnter element to Push");

scanf("%d",&item);

enqueue(item);

display();

break;

case 2:

printf("\nDequeue element is %d",dequeue());

display();

break;

case 3:

printf("\nElements of Queue is ");

display();

break;

default:

choice =4;

}

}while(choice!=4);

}

void enqueue(int ele){

if(isFull()==1){

printf("\nQueue overflow");

}

else if(front==-1){

rear++;

front++;

a[front]=ele;

}

else{

rear++;

a[rear]=ele;

}

}

int dequeue(){

int temp;

if(isEmpty()){

printf("\nunderflow");

}else if(front==rear){

temp=a[front];

rear=-1;

front=-1;

return temp;

}

else{

temp=a[front];

front++;

return temp;

}

}

int isEmpty(){

if(front==-1){

return 1;

}

return 0;

}

int isFull(){

if(rear==(size)-1){

return 1;

}

return 0;

}

void display(){

if(isEmpty()==1){

printf("\nQueue is empty");

}else{

for(i=front;i<=rear;i++){

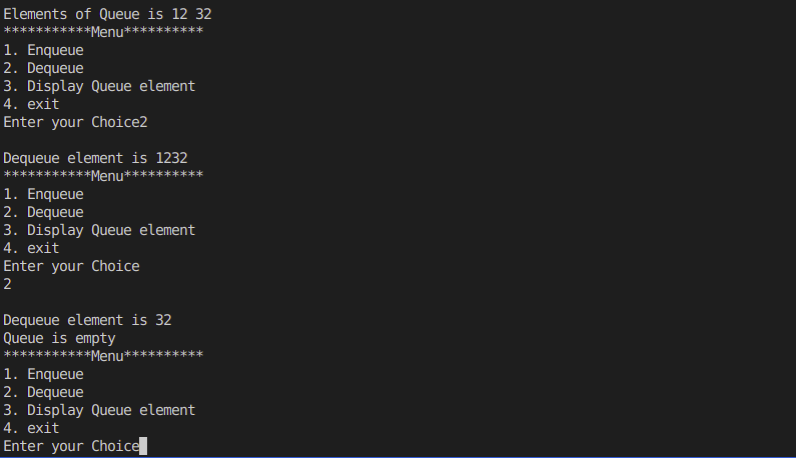
printf("%d ",a[i]);

}

}

}

Output:-



CP3.

#include<stdlib.h>

#include <stdio.h>

#define SIZE 5

int cir\_queue[SIZE];

int front = -1, rear = -1;

int isFull() {

if ((front == rear + 1) || (front == 0 && rear == SIZE - 1))

return 1;

return 0;

}

int isEmpty() {

if (front == -1)

return 1;

return 0;

}

void enqueue(int element) {

if (isFull())

printf("\n Queue is full!!! \n");

else {

if (front == -1) front = 0;

rear = (rear + 1) % SIZE;

cir\_queue [rear] = element;

printf("\n Inserted element: %d", element);

}

}

int dequeue() {

int element;

if (isEmpty()) {

printf("\n Queue is empty!!! \n");

return (-1);

} else {

element = cir\_queue [front];

if (front == rear) {

front = -1;

rear = -1;

}

else {

front = (front + 1) % SIZE;

}

printf("\n Deleted element: %d \n", element);

return (element);

}

}

void display() {

int i;

if (isEmpty())

printf(" \n Empty Queue\n");

else {

printf("\n Circular queue: ");

for (i = front; i != rear; i = (i + 1) % SIZE) {

printf("%d ", cir\_queue [i]);

}

printf("%d ", cir\_queue [i]);

}

}

int main() {

int ch,item;

printf("\nCoded by Deepak(04335304421)\n");

do

{

printf("\n\*\*\*\*\*\*\*\*\*\*OPTIONS\*\*\*\*\*\*\*\*\*\*");

printf("\n1.Enqueue ");

printf("\n2.Dequeue ");

printf("\n3.Display the Queue");

printf("\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\nElement to be inserted in the Queue : ");

scanf("%d", &item);

enqueue(item);

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("Incorrect choice \n");

}

}while(1);

return 0;

}

Output:-

