Name: Shah Vrushti Himanshubhai

Enrolment No.: ITD2D\_211622

Subject: Data Structure (3130702)

**Practical-1**

**C fundamentals**

1. **Implement a C program to demonstrate “swapping of two variables” (1) using a pointer (2)without using third variable**
2. **Using a pointer**

#include <stdio.h>

#include<conio.h>

int main()

{

int x, y, \*a, \*b, temp;

printf("Enter the value of x and y\n");

scanf("%d%d", &x, &y);

printf("Before Swapping\nx = %d\ny = %d\n", x, y);

a = &x;

b = &y;

temp = \*b;

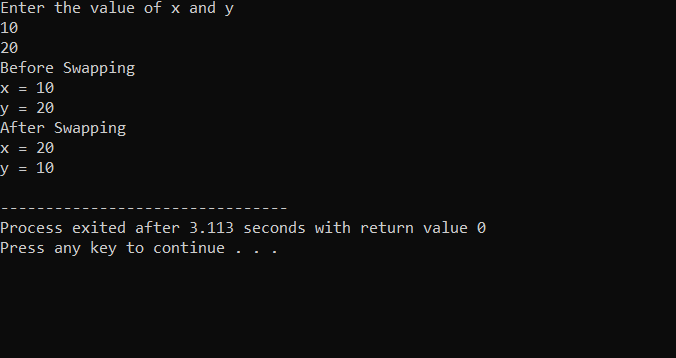
\*b = \*a;

\*a = temp;

printf("After Swapping\nx = %d\ny = %d\n", x, y);

return 0;

}



1. **Without using third variable**

#include<stdio.h>

#include<conio.h>

int main()

{

int a=10, b=20;

printf("Before swap a=%d b=%d",a,b);

a=a+b;

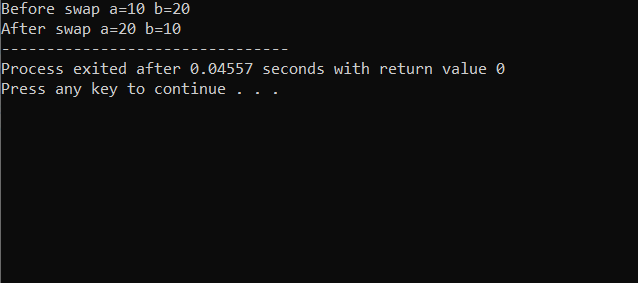
b=a-b;

a=a-b;

printf("\nAfter swap a=%d b=%d",a,b);

return 0;

}



1. **Create a structure named “Library” having the data members as “Book Id”, ”Book Title”, “Name of author”, “Total no of copies”, “Available number of copies”. Demonstrate the menu driven functionality which can perform following: (1) Display all records of library (2) Display all record of available books of library (3) Get the book id from the user and get the availability status in return.**
2. **Display all records of library**
3. **Display all record of available books of library**
4. **Get the book id from the user and get the availability status in return.**

#include<stdio.h>

#include<conio.h>

struct library

{

int book\_id;

char book\_title[50];

char authour[50];

int total\_copies;

int available\_copies;

}data[5];

void display(int r)

{

int i;

printf("All the records of library:\n");

printf("-----------------------------------------------------\n");

printf("Book\_ID\tBook\_title\tAuthour\tTotal\_copies\tAvailable\_copies\n");

printf("-----------------------------------------------------\n");

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

{

printf("%d\t%s\t%s\t%d\t%d\n",data[i].book\_id,data[i].book\_title,data[i].authour,data[i].total\_copies,data[i].available\_copies);

}

}

void available\_books(int r)

{

int i;

printf("Available books in library:\n");

printf("-----------------------------\n");

printf("Book\_ID\tBook\_title\tAvailable\_copies\n");

printf("-------------------------------------\n");

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

{

if(data[i].available\_copies>0)

{

printf("%d\t%s\t%d\n",data[i].book\_id,data[i].book\_title,data[i].available\_copies);

}

}

}

void status(int r)

{

int i,id;

printf("Enter book id for search the status:\n");

scanf("%d",&id);

printf("---------------------------------\n");

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

{

if(data[i].book\_id==id)

{

if(data[i].available\_copies>0)

printf("Book is available");

else

printf("Book is not available");

}

}

}

int main()

{

int i,n,ch;

printf("How many records you want to add:");

scanf("%d",&n);

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

{

printf("\nEnter details of library books:%d",i+1);

printf("\n");

printf("Enter book ID:");

scanf("%d",&data[i].book\_id);

printf("Enter book title:");

scanf("%s",&data[i].book\_title);

printf("Enter authour of book :");

scanf("%s",&data[i].authour);

printf("Enter total copies of book:");

scanf("%d",&data[i].total\_copies);

printf("Enter available copies of book:");

scanf("%d",&data[i].available\_copies);

}

while(1)

{

printf("\nMenu");

printf("\n 1.Display all data of library \n 2.Display available books of library \n 3.Enter your book id and check availability status \n 4.Exit");

printf("\n Enter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

display(n);

break;

case 2:

available\_books(n);

break;

case 3:

status(n);

break;

case 4:

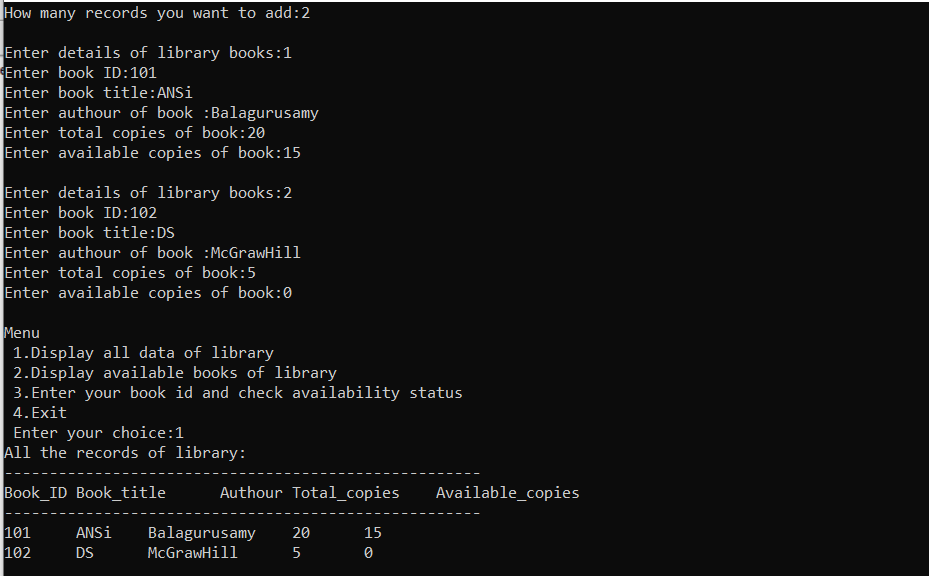
exit(0);

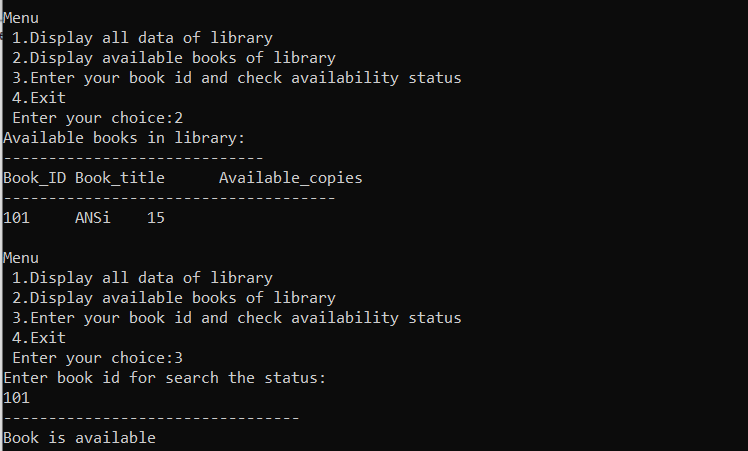
}

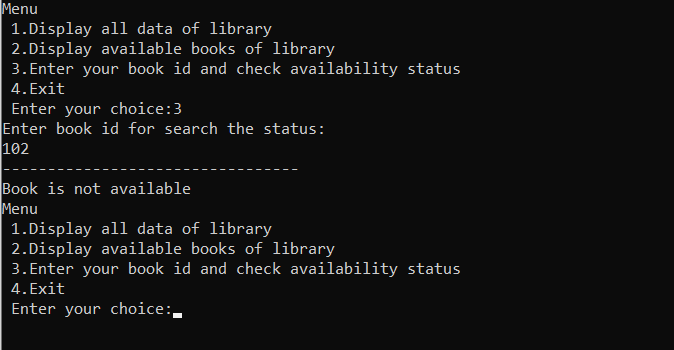
}

return 0;

}







1. **Create a code of your choice which demonstrate the usage of following function: (1) malloc (2) calloc (3) realloc**
2. **Malloc & Calloc**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int \*ptr, \*ptr1;

int n, i;

n = 5;

printf("Enter number of elements: %d\n", n);

ptr = (int\*)malloc(n \* sizeof(int));

ptr1 = (int\*)calloc(n, sizeof(int));

if (ptr == NULL || ptr1 == NULL) {

printf("Memory not allocated.\n");

exit(0);

}

else {

printf("Memory successfully allocated using malloc.\n");

free(ptr);

printf("Malloc Memory successfully freed.\n");

printf("\nMemory successfully allocated using calloc.\n");

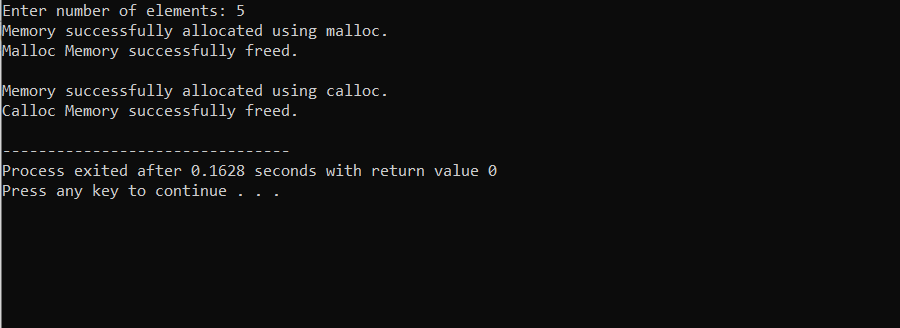
free(ptr1);

printf("Calloc Memory successfully freed.\n");

}

return 0;

}



1. **Realloc**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int\* ptr;

int n, i;

n = 5;

printf("Enter number of elements: %d\n", n);

ptr = (int\*)calloc(n, sizeof(int));

if (ptr == NULL) {

printf("Memory not allocated.\n");

exit(0);

}

else {

printf("Memory successfully allocated using calloc.\n");

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

ptr[i] = i + 1;

}

printf("The elements of the array are: ");

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

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

}

n = 10;

printf("\n\nEnter the new size of the array: %d\n", n);

ptr = realloc(ptr, n \* sizeof(int));

printf("Memory successfully re-allocated using realloc.\n");

for (i = 5; i < n; ++i) {

ptr[i] = i + 1;

}

printf("The elements of the array are: ");

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

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

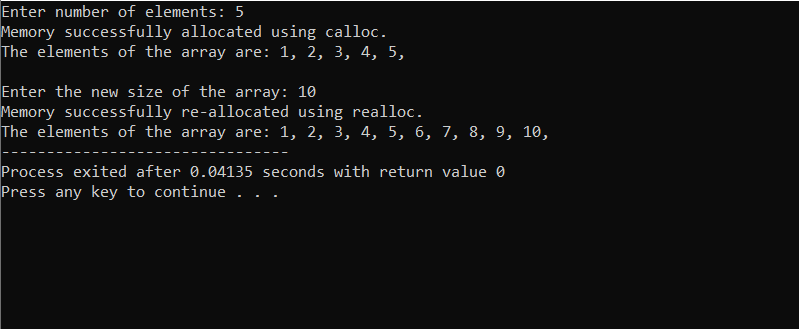
}

free(ptr);

}

return 0;

}



**Practical-2**

**Working with Arrays**

**Given an integer array of size 20 which is empty initially. On the given array, implement the following functions for it.**

1. **Insert: Get a number from a user to be inserted and insert on to the top.**
2. **Delete: Remove the top most element**
3. **Search: Get a number from a user to be searched and return its location**
4. **Sort: Sort all the numbers in ascending/descending order**
5. **Traverse: Display all the elements of an array**
6. **Insert:**

#include<stdio.h>>

int main()

{

int array[20], position, c, n, value;

printf("\n\nEnter number of elements in array:");

scanf("%d", &n);

printf("\n\nEnter %d elements\n", n);

for(c = 0; c < n; c++)

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

printf("\n\nEnter the location where you want to insert new element: ");

scanf("%d", &position);

printf("\n\nEnter the value to insert: ");

scanf("%d", &value);

for(c = n-1; c >= position-1; c--)

array[c+1] = array[c];

array[position - 1] = value;

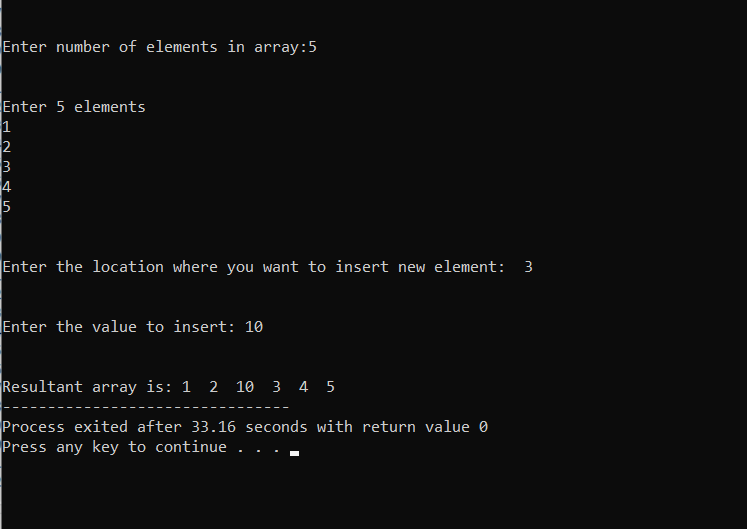
printf("\n\nResultant array is: ");

for(c = 0; c <= n; c++)

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

return 0;

}



1. **Delete:**

#include<stdio.h>

int main()

{

int array[20], position, c, n;

printf("\n\nEnter number of elements in array:");

scanf("%d", &n);

printf("\n\nEnter %d elements\n", n);

for(c = 0; c < n; c++)

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

printf("\n\nEnter the location where you want to delete element from: ");

scanf("%d", &position);

if(position >= n+1)

printf("\n\nDeletion not possible\n\n");

else

for(c = position-1; c < n-1; c++)

array[c] = array[c+1];

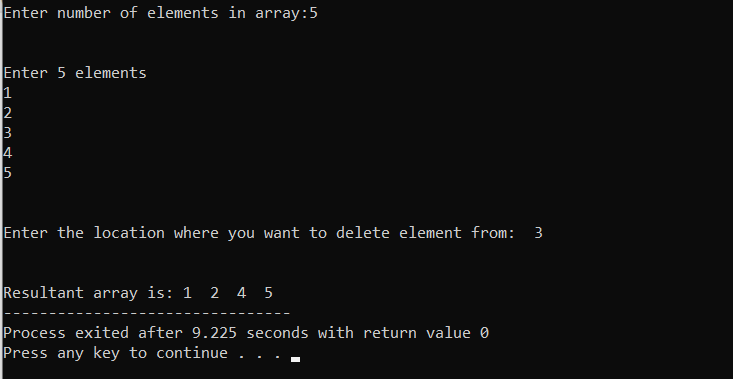
printf("\n\nResultant array is: ");

for(c = 0; c < n-1; c++)

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

return 0;

}



1. **Search:**

#include <conio.h>

int main()

{

int a[20],i,n,key;

printf("Enter size of the array : ");

scanf("%d", &n);

printf("Enter elements in array : ");

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

{

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

}

printf("Enter the element to search : ");

scanf("%d", &key);

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

{

if(a[i]==key)

{

printf("element found ");

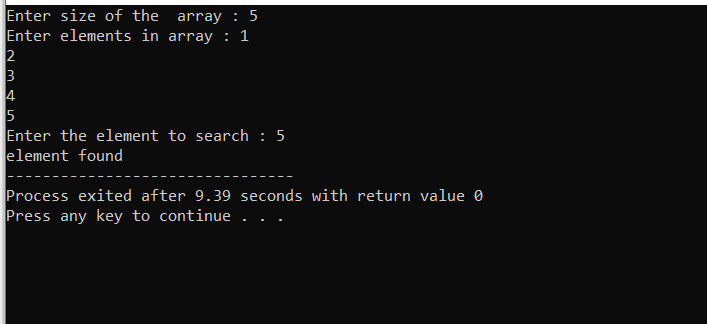
return 0;

}

}

printf("element not found");

}



1. **Sort:**

#include <stdio.h>

void main()

{

int arr1[100];

int n, i, j, tmp;

printf("\n\nsort elements of array in ascending order :\n ");

printf("----------------------------------------------\n");

printf("Input the size of array : ");

scanf("%d", &n);

printf("Input %d elements in the array :\n",n);

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

{

printf("element - %d : ",i);

scanf("%d",&arr1[i]);

}

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

{

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

{

if(arr1[j] <arr1[i])

{

tmp = arr1[i];

arr1[i] = arr1[j];

arr1[j] = tmp;

}

}

}

printf("\nElements of array in sorted ascending order:\n");

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

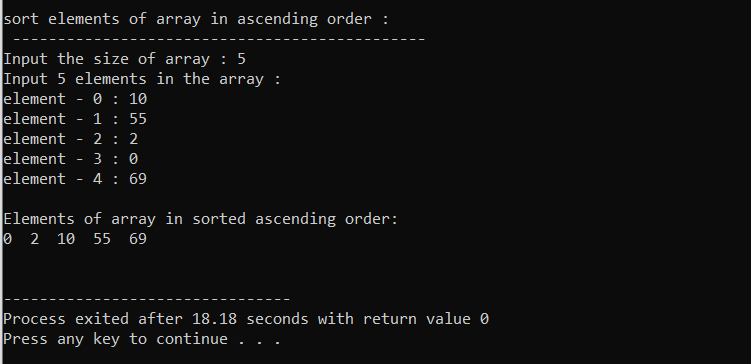
{

printf("%d ", arr1[i]);

}

printf("\n\n");

}



1. **Traverse:**

#include <stdio.h>

void printArray(int\* arr, int n)

{

int i;

printf("Array: ");

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

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

}

printf("\n");

}

int main()

{

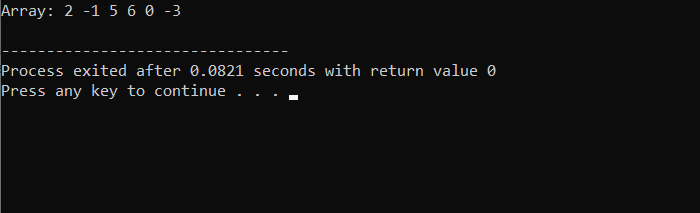
int arr[] = { 2, -1, 5, 6, 0, -3 };

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

printArray(arr, n);

return 0;

}



**Practical-3**

**Working with Stack-basics**

1. **Given an integer stack of size 20 which is empty initially. On the given stack, implement the following functions for it.**
2. **PUSH**
3. **POP**
4. **PEEP**
5. **Change**
6. **Display**

#include<stdio.h>

#include<conio.h>

#define n 30

int top=-1,s[n]; //top is a pointer and save values of index

void push();

void pop();

void peek();

void display();

void search();

void peep();

void change();

void main()

{

int ch;

while(1)

{

printf("\n\nStack Menu:\n\n1.Push \n2.Pop \n3.Peek \n4.Display or traverse \n5.Search \n6.Peep \n7.Change \n\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5:

search();

break;

case 6:

peep();

break;

case 7:

change();

break;

}

}

}

void push()

{

int val;

if(top==n-1)

printf("\nStack is Overflow");

else

{

printf("\nEnter element to push");

scanf("%d",&val);

top=top+1;

s[top]=val;

}

display();

}

void pop()

{

if(top==-1)

printf("\nStack is underflow");

else

{

printf("\nDeleted element is %d",s[top]);

top=top-1;

}

}

void peek()

{

if(top==-1)

printf("\nStack is underflow");

else

{

printf("\nPeek or top element of the stack is:%d",s[top]);

}

}

void display()

{

int i;

if(top==-1)

{

printf("\nStack is underflow");

}

else

{

printf("\n stack is:");

for(i=top; i>=0; i--)

{

printf("\n %d",s[i]);

}

}

}

void search()

{

int i,search,pos=-1;

if(top==-1)

{

printf("\nStack is underflow");

}

else

{

printf("Enter the element you want to search:");

scanf("%d",&search);

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

{

if(search==s[i])

{

pos=i+1;

break;

}

}

if(pos==-1)

{

printf("Element not found");

}

else

{

printf("Element found at position:%d",pos);

}

}

}

void peep()

{

int i;

printf("Enter positon for search:");

scanf("%d",&i);

if(top-i<=-1)

{

printf("Stack is underflow");

}

else

{

printf("The element is:%d",s[top-i+1]);

}

}

void change()

{

int i1,i2;

printf("Enter position for change element:");

scanf("%d",&i1);

printf("Enter number for update:");

scanf("%d",&i2);

if(top-i1<=-1)

{

printf("Stack is underflow");

}

else

{

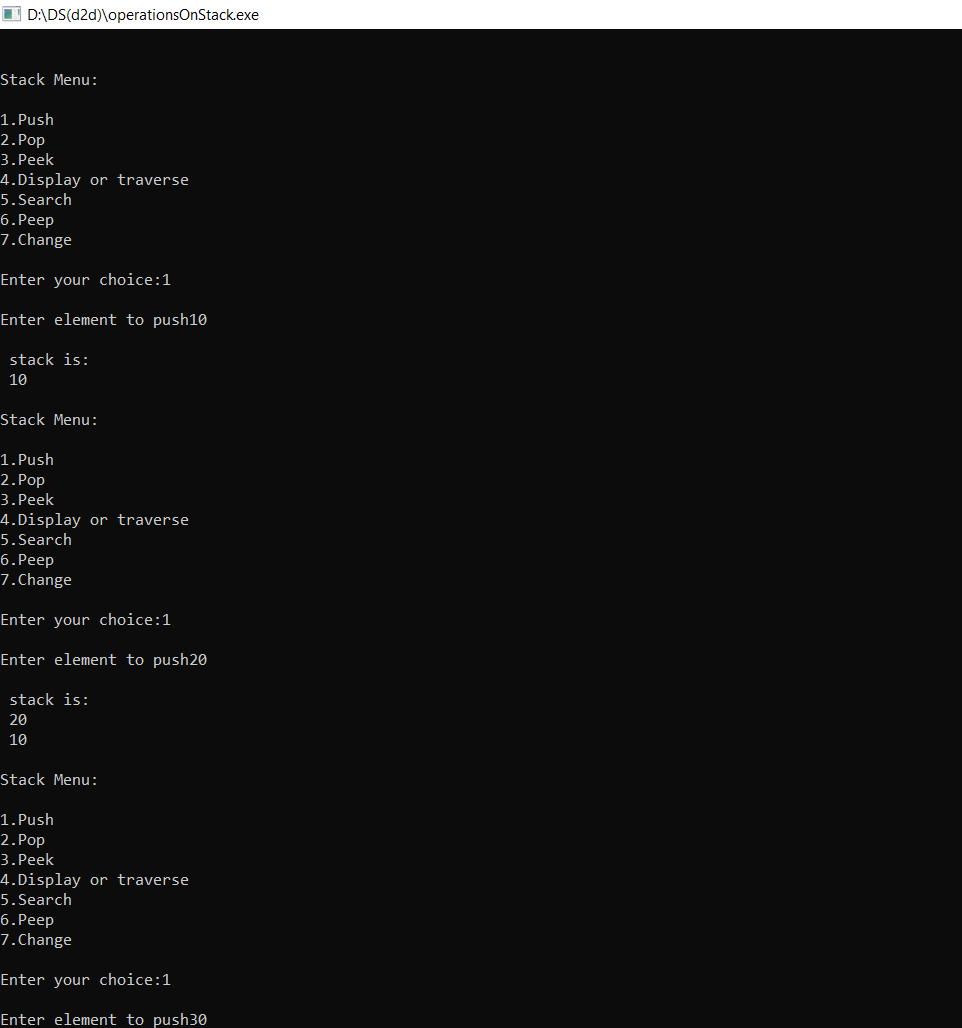
s[top-i1+1]=i2;

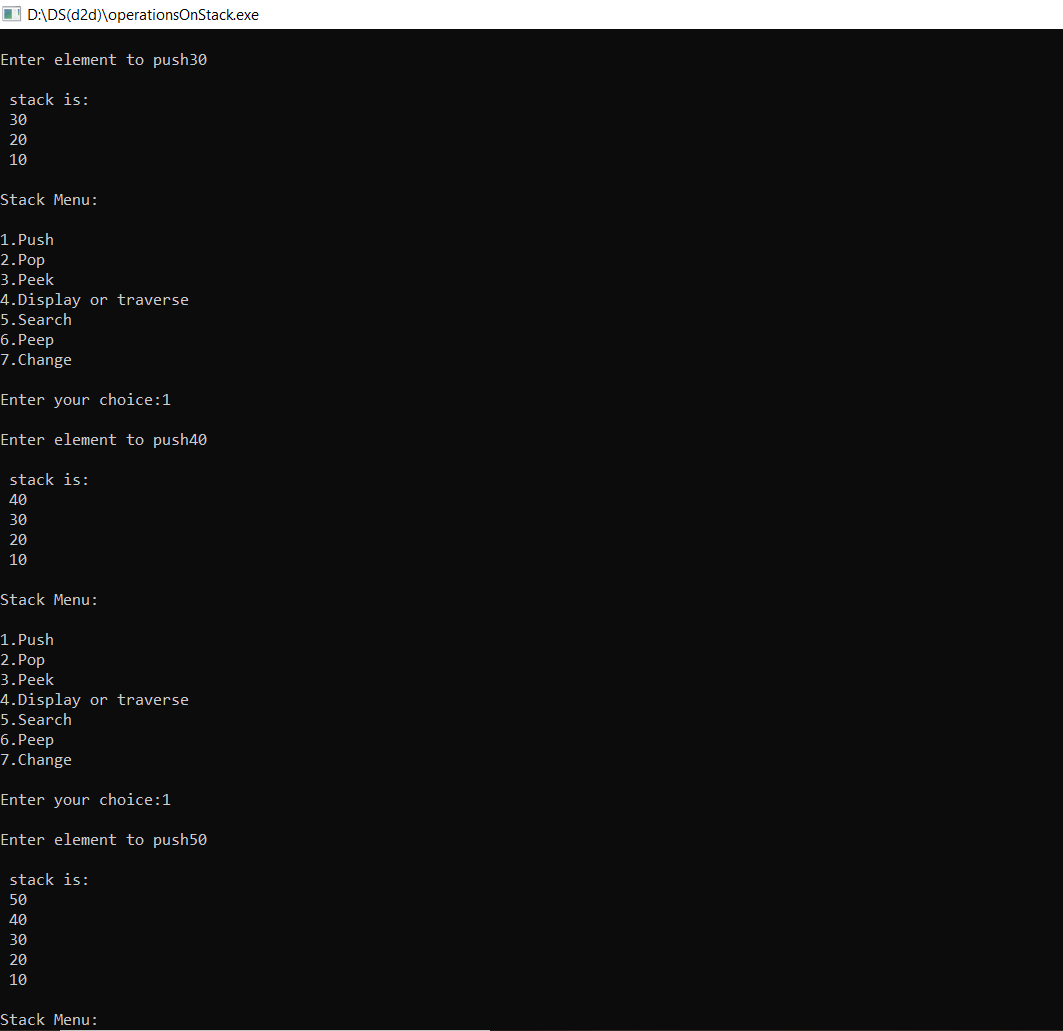
printf("Update successfully");

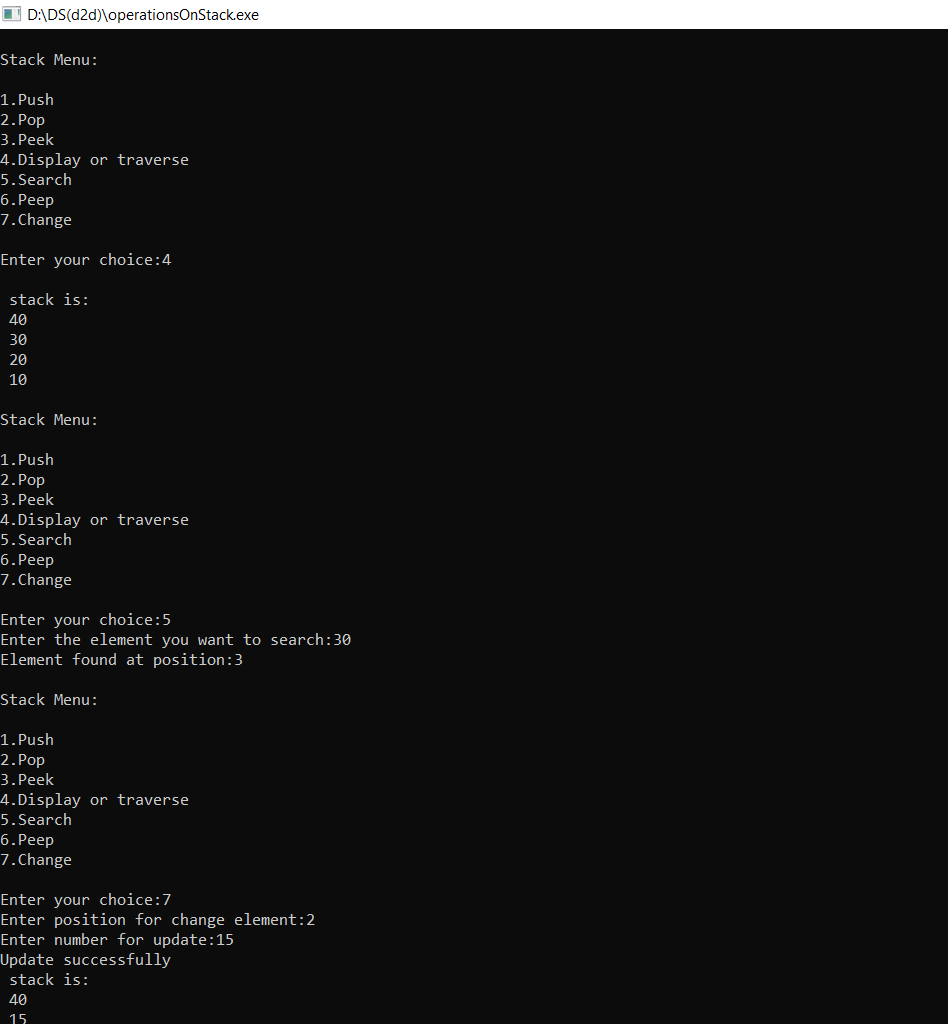
}

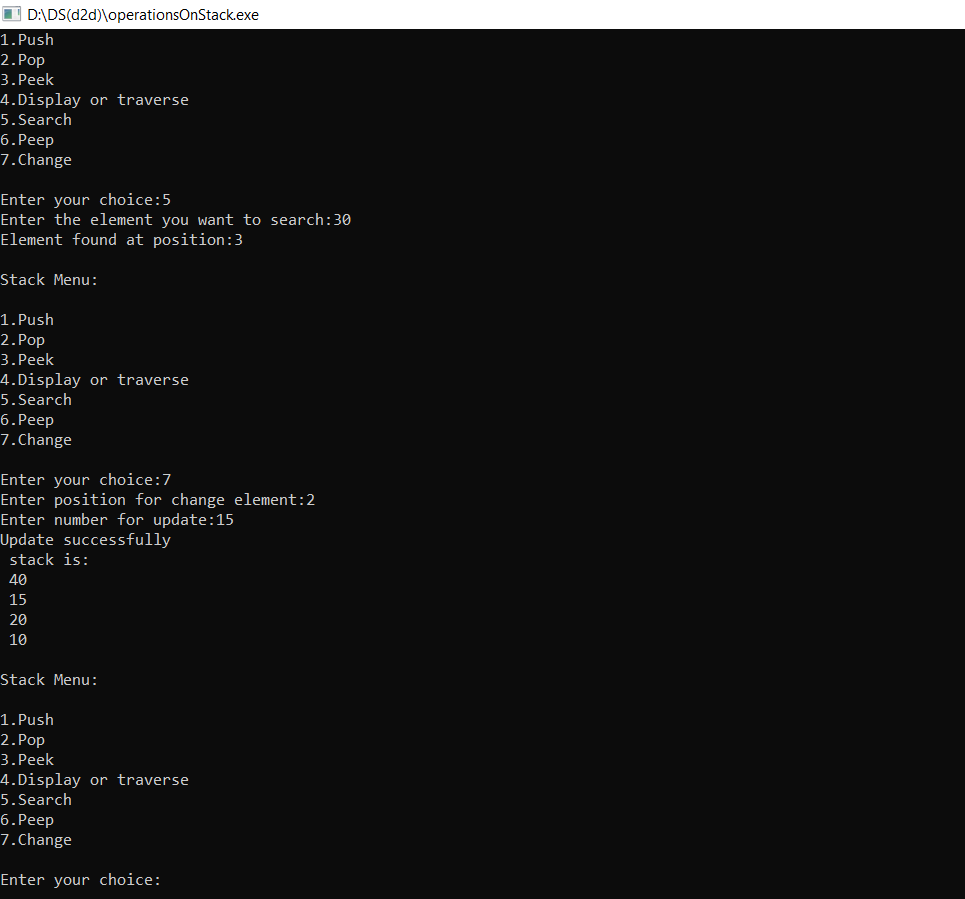
display();

}









1. **Implement two stacks in a single array of size 30. Demonstrate all the functions of stacks for both the stacks.**

#include <stdio.h>

#define SIZE 30

int ar[SIZE];

int top1 = -1;

int top2 = SIZE;

void push\_stack1 (int data)

{

if (top1 < top2 - 1)

{

ar[++top1] = data;

}

else

{

printf ("Stack Full! Cannot Push\n");

}

}

void push\_stack2 (int data)

{

if (top1 < top2 - 1)

{

ar[--top2] = data;

}

else

{

printf ("Stack Full! Cannot Push\n");

}

}

void pop\_stack1 ()

{

if (top1 >= 0)

{

int popped\_value = ar[top1--];

printf ("%d is being popped from Stack 1\n", popped\_value);

}

else

{

printf ("Stack Empty! Cannot Pop\n");

}

}

void pop\_stack2 ()

{

if (top2 < SIZE)

{

int popped\_value = ar[top2++];

printf ("%d is being popped from Stack 2\n", popped\_value);

}

else

{

printf ("Stack Empty! Cannot Pop\n");

}

}

void print\_stack1 ()

{

int i;

for (i = top1; i >= 0; --i)

{

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

}

printf ("\n");

}

void print\_stack2 ()

{

int i;

for (i = top2; i < SIZE; ++i)

{

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

}

printf ("\n");

}

int main()

{

int ar[SIZE];

int i;

int num\_of\_ele;

printf ("We can push a total of 10 values\n");

for (i = 1; i <= 6; ++i)

{

push\_stack1 (i);

printf ("Value Pushed in Stack 1 is %d\n", i);

}

for (i = 1; i <= 4; ++i)

{

push\_stack2 (i);

printf ("Value Pushed in Stack 2 is %d\n", i);

}

print\_stack1 ();

print\_stack2 ();

printf ("Pushing Value in Stack 1 is %d\n", 11);

push\_stack1 (11);

num\_of\_ele = top1 + 1;

while (num\_of\_ele)

{

pop\_stack1 ();

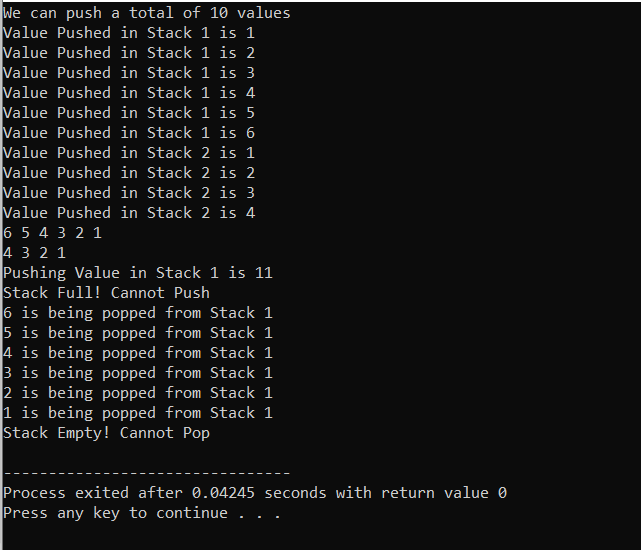
--num\_of\_ele;

}

pop\_stack1 ();

return 0;

}



**Practical-4**

**Working with Stack-Applications**

1. **Create a C code for converting given infix expression to postfix expression using stack.**

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define SIZE 100

char stack[SIZE];

int top = -1;

/\* define push operation \*/

void push(char item)

{

if(top >= SIZE-1)

{

printf("\nStack Overflow.");

}

else

{

top = top+1;

stack[top] = item;

}

}

/\* define pop operation \*/

char pop()

{

char item ;

if(top <0)

{

printf("stack under flow: invalid infix expression");

getchar();

exit(1);

}

else

{

item = stack[top];

top = top-1;

return(item);

}

}

int is\_operator(char symbol)

{

if(symbol == '^' || symbol == '\*' || symbol == '/' || symbol == '+' || symbol =='-')

{

return 1;

}

else

{

return 0;

}

}

int precedence(char symbol)

{

if(symbol == '^')/\* exponent operator, highest precedence\*/

{

return(3);

}

else if(symbol == '\*' || symbol == '/')

{

return(2);

}

else if(symbol == '+' || symbol == '-') /\* lowest precedence \*/

{

return(1);

}

else

{

return(0);

}

}

void InfixToPostfix(char infix\_exp[], char postfix\_exp[])

{

int i, j;

char item;

char x;

push('('); /\* push '(' onto stack \*/

strcat(infix\_exp,")"); /\* add ')' to infix expression \*/

i=0;

j=0;

item=infix\_exp[i]; /\* initialize before loop\*/

while(item != '\0') /\* run loop till end of infix expression \*/

{

if(item == '(')

{

push(item);

}

else if( isdigit(item) || isalpha(item))

{

postfix\_exp[j] = item;

j++;

}

else if(is\_operator(item) == 1)

{

x=pop();

while(is\_operator(x) == 1 && precedence(x)>= precedence(item))

{

postfix\_exp[j] = x;

j++;

x = pop();

}

push(x);

push(item);

}

else if(item == ')')

{

x = pop();

while(x != '(')

{

postfix\_exp[j] = x;

j++;

x = pop();

}

}

else

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

i++;

item = infix\_exp[i];

}

if(top>0)

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

if(top>0)

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

postfix\_exp[j] = '\0';

}

int main()

{

char infix[SIZE], postfix[SIZE];

printf("ASSUMPTION: The infix expression contains single letter variables and single digit constants only.\n");

printf("\nEnter Infix expression : ");

gets(infix);

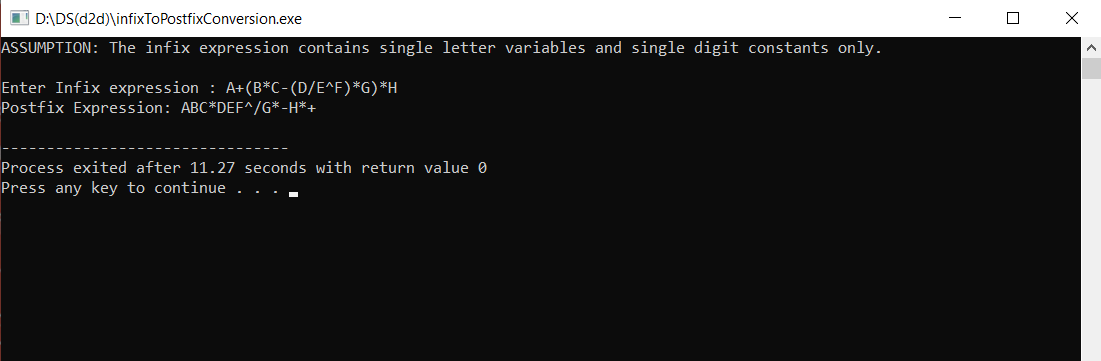
InfixToPostfix(infix,postfix);

printf("Postfix Expression: ");

puts(postfix);

return 0;

}



1. **Given an expression as “46+64-/”, evaluate the expression using the stack.**

#include <stdio.h>

#include <ctype.h>

#define MAXSTACK 100

#define POSTFIXSIZE 100

int stack[MAXSTACK];

int top = -1;

void push(int item)

{

if (top >= MAXSTACK - 1) {

printf("stack over flow");

return;

}

else {

top = top + 1;

stack[top] = item;

}

}

int pop()

{

int item;

if (top < 0) {

printf("stack under flow");

}

else {

item = stack[top];

top = top - 1;

return item;

}

}

void EvalPostfix(char postfix[])

{

int i;

char ch;

int val;

int A, B;

for (i = 0; postfix[i] != ')'; i++) {

ch = postfix[i];

if (isdigit(ch)) {

push(ch - '0');

}

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {

A = pop();

B = pop();

switch (ch)

{

case '\*':

val = B \* A;

break;

case '/':

val = B / A;

break;

case '+':

val = B + A;

break;

case '-':

val = B - A;

break;

}

push(val);

}

}

printf(" \n Result of expression evaluation : %d \n", pop());

}

int main()

{

int i;

char postfix[POSTFIXSIZE];

printf("ASSUMPTION: There are only four operators(\*, /, +, -) in an expression and operand is single digit only.\n");

printf(" \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

for (i = 0; i <= POSTFIXSIZE - 1; i++) {

scanf("%c", &postfix[i]);

if (postfix[i] == ')') /\* is there any way to eliminate this if \*/

{

break;

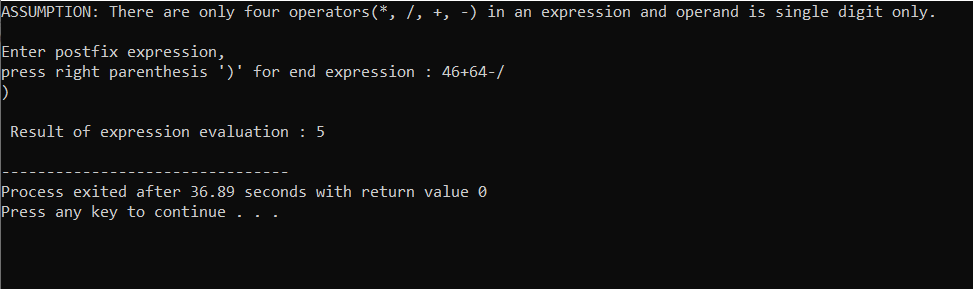
}

}

EvalPostfix(postfix);

return 0;

}



**Practical-5**

**Working with Queue-Basics**

1. **Given an integer array of size 20 which is empty initially. On the given array, implement the following functions such that it can functions as a queue, which should be able to perform the following operations.**
2. **Insert(Enqueue)**
3. **Delete(Dequeue)**
4. **Display**

#include<stdio.h>

#include<conio.h>

#define MAX 5

void enqueue();

void dequeue();

void display();

int rear=-1;

int front=-1;

int q[MAX];

main()

{

int c;

while(1)

{

printf("\nChoices of operation:\n 1.Enqueue(Insert) \n 2.Dequeue(Delete) \n 3.Display");

printf("\nEnter your choice:");

scanf("%d",&c);

switch(c)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

}

}

}

void enqueue()

{

int val;

if(rear==MAX-1)

printf("Queue is overflow");

else

{

if(front==-1)

front=0;

printf("Enter value for insert:");

scanf("%d",&val);

rear=rear+1;

q[rear]=val;

}

display();

}

void dequeue()

{

if(front==-1 && rear==-1)

printf("Queue is underflow");

else if(rear==front)

{

rear=-1;

front=-1;

}

else

{

printf("\nDeleted element is:%d",q[front]);

front=front+1;

}

}

void display()

{

int i;

if(front==-1)

printf("Queue is empty");

else

{

printf("Queue is:");

printf("\n");

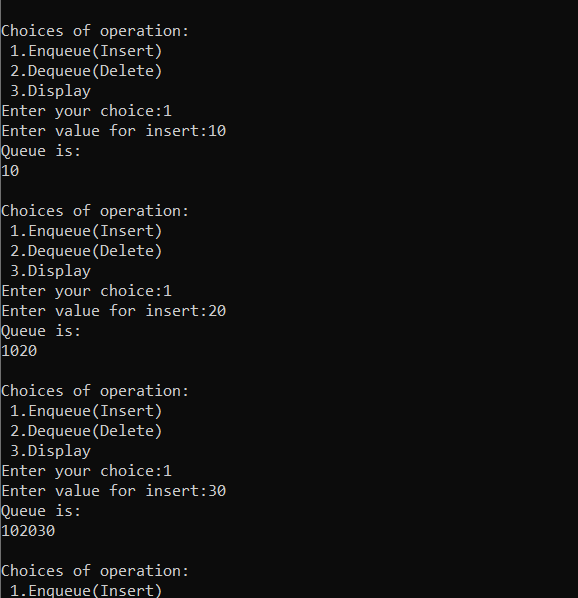
for(i=front;i<=rear;i++)

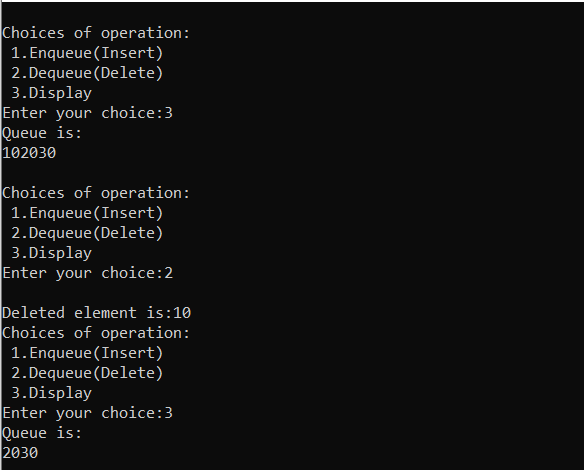
printf("%d",q[i]);

}

printf("\n");

}





**Practical-6**

**Working with Queue-Advanced**

1. **Create a C program to** **record the dummy browser history where the array is assumed to store the web pages user manually enters as the page visited. Consider the size of the array is 10.**

**Expected Output:**

**Select from below:**

1. **Add page of visit.**
2. **Display visited pages**

**Choice: 1**

**Which page to add for visit?:** [**www.google.com**](http://www.google.com)

**Do you want to continueue? Y**

**Select from below:**

1. **Add page of visit.**
2. **Display visited pages**

**Choice:1**

**Which page add for visit?:** [**www.facebook.com**](http://www.facebook.com)

**Do you want to continueue? Y**

**Select from below:**

1. **Add page of visit.**
2. **Display visited pages**

**Choice:2**

**The pages visited are**

[**www.facebook.com**](http://www.facebook.com)

[**www.google.com**](http://www.google.com)

**Practical-7**

**Working with Circular Queue-Basics**

1. **Write a program to implement Circular queue using arrays that performs following operations.**
2. **INSERT**
3. **DELETE**
4. **DISPLAY**

#include<stdio.h>

# define MAX 5

int cqueue\_arr[MAX];

int front = -1;

int rear = -1;

void insert(int item)

{

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

{

printf("\nQueue Overflow");

return;

}

if(front == -1)

{

front = 0;

rear = 0;

}

else

{

if(rear == MAX-1)

rear = 0;

else

rear = rear+1;

}

cqueue\_arr[rear] = item ;

display();

}

void deletion()

{

if(front == -1)

{

printf("\nQueue Underflow");

return ;

}

printf("\nElement deleted from queue is : %d",cqueue\_arr[front]);

if(front == rear)

{

front = -1;

rear=-1;

}

else

{

if(front == MAX-1)

front = 0;

else

front = front+1;

}

display();

}

void display()

{

int front\_pos = front,rear\_pos = rear;

if(front == -1)

{

printf("\nQueue is empty");

return;

}

printf("\nQueue elements :");

if( front\_pos <= rear\_pos )

while(front\_pos <= rear\_pos)

{

printf("%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

else

{

while(front\_pos <= MAX-1)

{

printf("%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

front\_pos = 0;

while(front\_pos <= rear\_pos)

{

printf("%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

}

printf("\n");

}

int main()

{

int choice,item;

do

{

printf("\n1.Insert");

printf("\n2.Delete");

printf("\n3.Display");

printf("\n4.Quit");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1 :

printf("\nInput the element for insertion in queue : ");

scanf("%d", &item);

insert(item);

break;

case 2 :

deletion();

break;

case 3:

display();

break;

case 4:

break;

default:

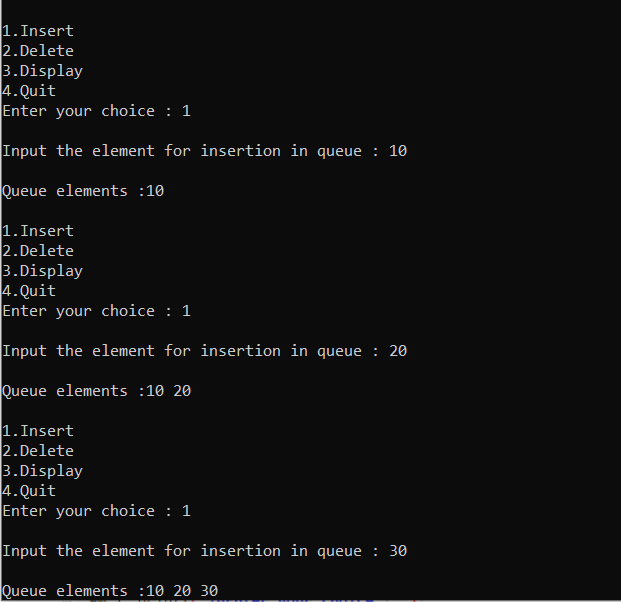
printf("Wrong choicen");

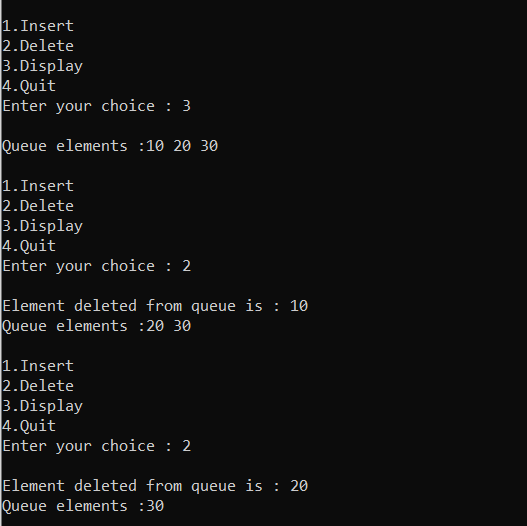
}

}while(choice!=4);

return 0;

}





**Extra Programs**

1. **Tower of Hanoi.**

#include<stdio.h>

#include<conio.h>

int toh(int a,char from\_rod,char to\_rod,char extra\_rod)

{

if(a==1)

{

printf("\nMove disk 1 from %c to %c",from\_rod,to\_rod);

return;

}

toh(a-1,from\_rod,extra\_rod,to\_rod);

printf("\nMove disk %d from %c to %c",a,from\_rod,to\_rod);

toh(a-1,extra\_rod,to\_rod,from\_rod);

}

void main()

{

int n;

printf("Enter no. of disks you want to transfer:");

scanf("%d",&n);

toh(n,'A','B','C');

}

