Q)Program to represent Linked Stacks

**PROGRAM CODE**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

typedef struct node node;

node \*head;

void push()

{

int num;

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

if (ptr == NULL)

{

printf("not able to push the element");

}

else

{

printf("Enter the value: ");

scanf("%d", &num);

if (head == NULL)

{

ptr->num = num;

ptr->next = NULL;

head = ptr;

}

else

{

ptr->num = num;

ptr->next = head;

head = ptr;

}

printf("Item pushed");

}

}

void pop()

{

int item;

node \*ptr;

if (head == NULL)

{

printf("Underflow");

}

else

{

item = head->num;

ptr = head;

head = head->next;

free(ptr);

printf("Item popped");

}

}

void disp()

{

int i;

node \*ptr;

ptr = head;

if (ptr == NULL)

{

printf("Stack is empty\n");

}

else

{

printf("Printing Stack elements: ");

while (ptr != NULL)

{

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

ptr = ptr->next;

}

}

}

void main()

{

int ch;

while (1)

{

printf("\n\nMENU");

printf("\n1.Push\n2.Pop\n3.Display\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 1: push();

break;

case 2: pop();

break;

case 3: disp();

break;

case 4: exit(0);

break;

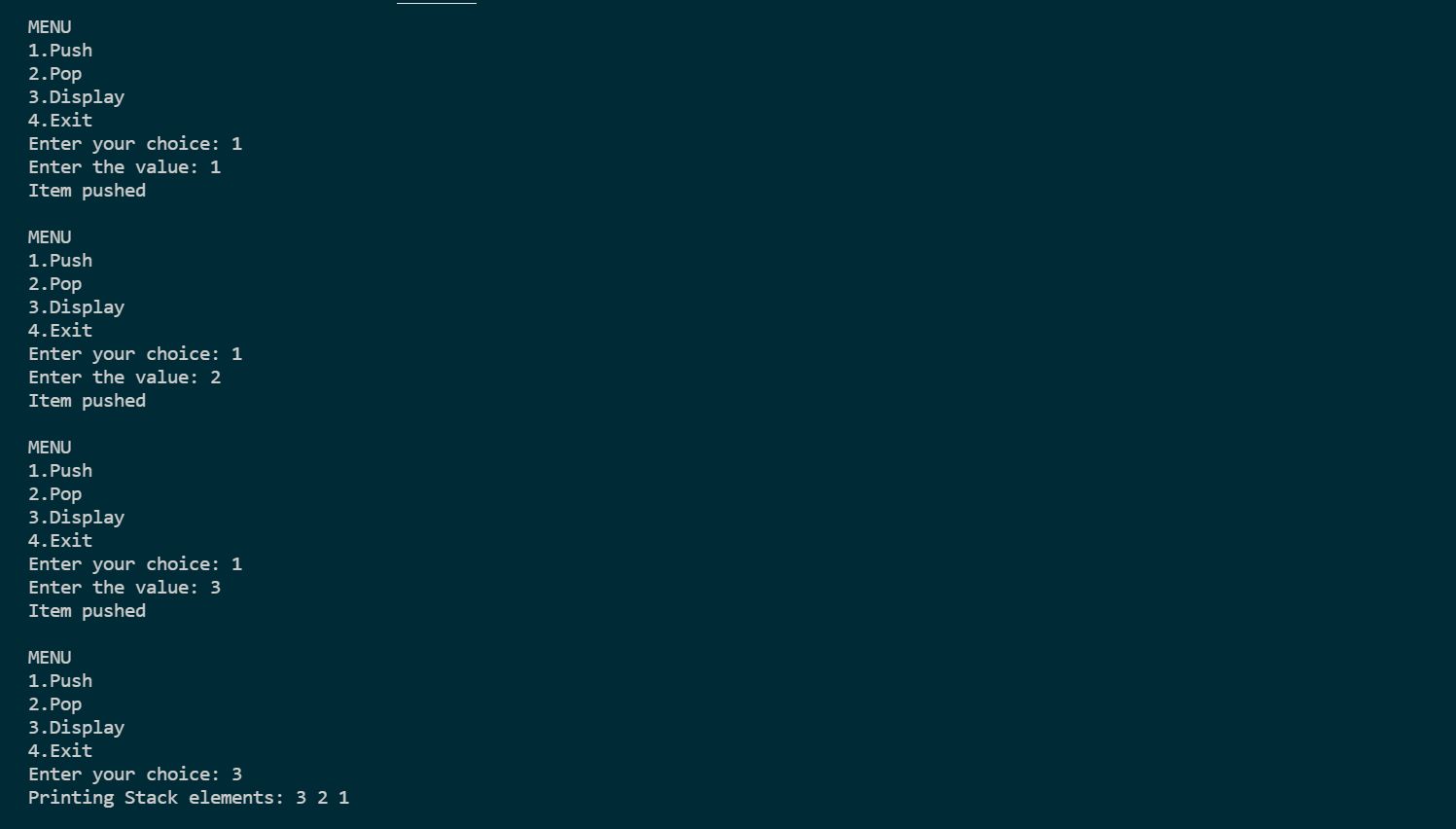
default: printf("\nINVALID INPUT");

}

}

}

**OUTPUT**



Q)Program to represent Linked Queue

**PROGRAM CODE**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

typedef struct node node;

node \*front, \*rear;

void ins()

{

struct node \*ptr;

int item;

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

if (ptr == NULL)

{

printf("OVERFLOW");

return;

}

else

{

printf("Enter value:");

scanf("%d", &item);

ptr->data = item;

if (front == NULL)

{

front = ptr;

rear = ptr;

front->next = NULL;

rear->next = NULL;

}

else

{

rear->next = ptr;

rear = ptr;

rear->next = NULL;

}

}

}

void del()

{

node \*ptr;

if (front == NULL)

{

printf("UNDERFLOW");

return;

}

else

{

ptr = front;

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

front = front->next;

free(ptr);

}

}

void disp()

{

node \*ptr;

ptr = front;

if (front == NULL)

{

printf("Empty queue");

}

else

{

printf("Printing Values: ");

while (ptr!= NULL)

{

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

ptr = ptr->next;

}

}

}

void main()

{

int ch;

while (1)

{

printf("\n\nMENU");

printf("\n1.Push\n2.Pop\n3.Display\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 1: ins();

break;

case 2: del();

break;

case 3: disp();

break;

case 4: exit(0);

break;

default:

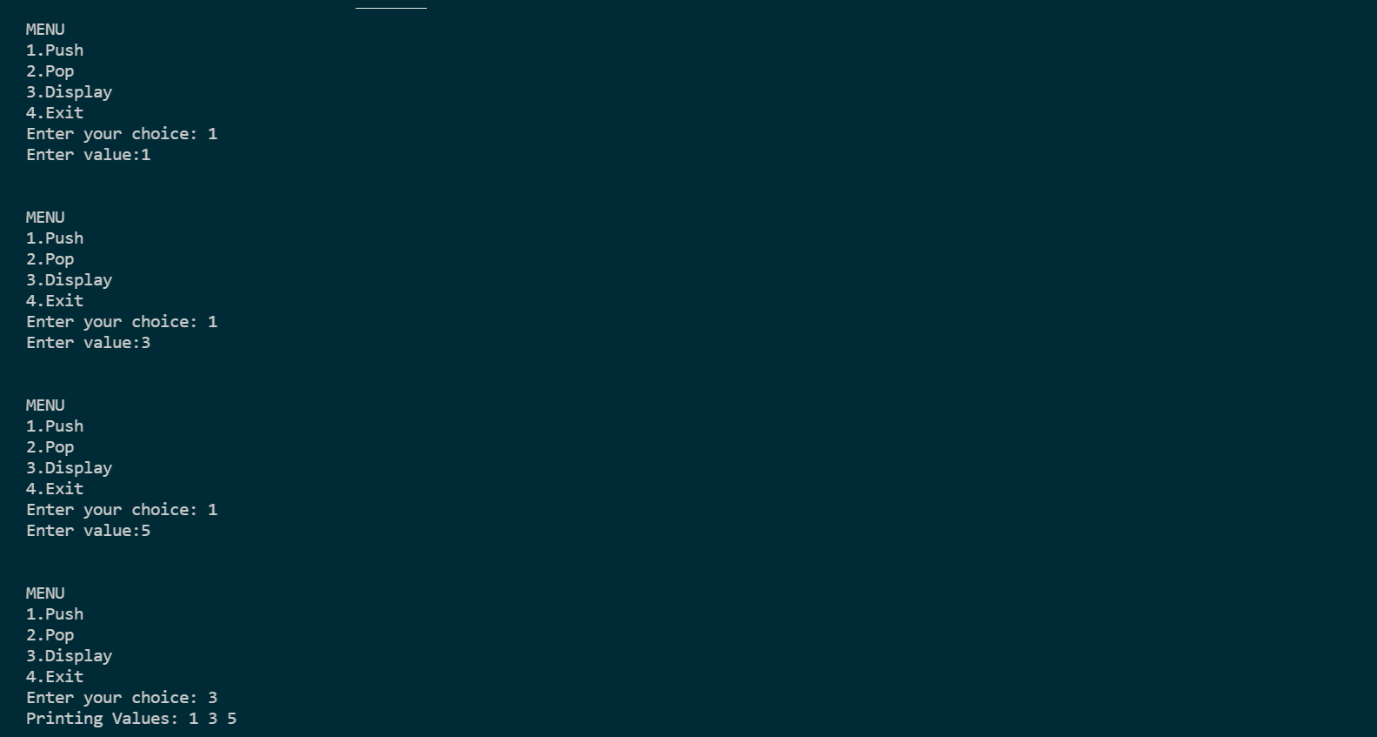
printf("\nINVALID INPUT");

}

}

}

**OUTPUT**



Q) Doubly Linked List

**PROGRAM CODE**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*prev,\*next;

};

typedef struct node node;

node \*head=NULL,\*tail=NULL,\*temp;

void ins(){

int n,ch;

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

printf("Enter the value: ");

scanf("%d",&n);

temp->data=n;

printf("1)Insert at beggining\n2)Insert at end\nChoice: ");

scanf("%d",&ch);

if(head==NULL)

{

temp->prev=NULL;

temp->next=NULL;

head=tail=temp;

}

else

{

if(ch==1){

temp->next=head;

head->prev=temp;

temp->prev=NULL;

head=temp;

}

else if(ch==2){

tail->next=temp;

temp->prev=tail;

temp->next=NULL;

tail=temp;

}

else

printf("Invalid Input");

}

}

void del()

{

int ch;

if (head == NULL)

printf("The list is empty");

else if (head->next==NULL)

{

temp=head;

free(temp);

head = tail = NULL;

}

else

{

printf("1)Delete from beggining\n2)Delete from end\nChoice: ");

scanf("%d", &ch);

if (ch == 1)

{

temp = head;

head = head->next;

tail->prev = NULL;

free(temp);

}

else if (ch == 2)

{

temp=tail;

tail=tail->prev;

free(temp);

}

else

printf("INVALID INPUT");

}

}

void disp()

{

if(head==NULL)

printf("No elements");

else

{

printf("The elements are: ");

for(temp=head;temp!=NULL;temp=temp->next)

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

}

}

void main(){

int ch;

while(1){

printf("\n\nMENU\n1.Insert\n2.Delete\n3.Display\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d",&ch);

switch(ch){

case 1: ins();

break;

case 2: del();

break;

case 3: disp();

break;

case 4: exit(0);

break;

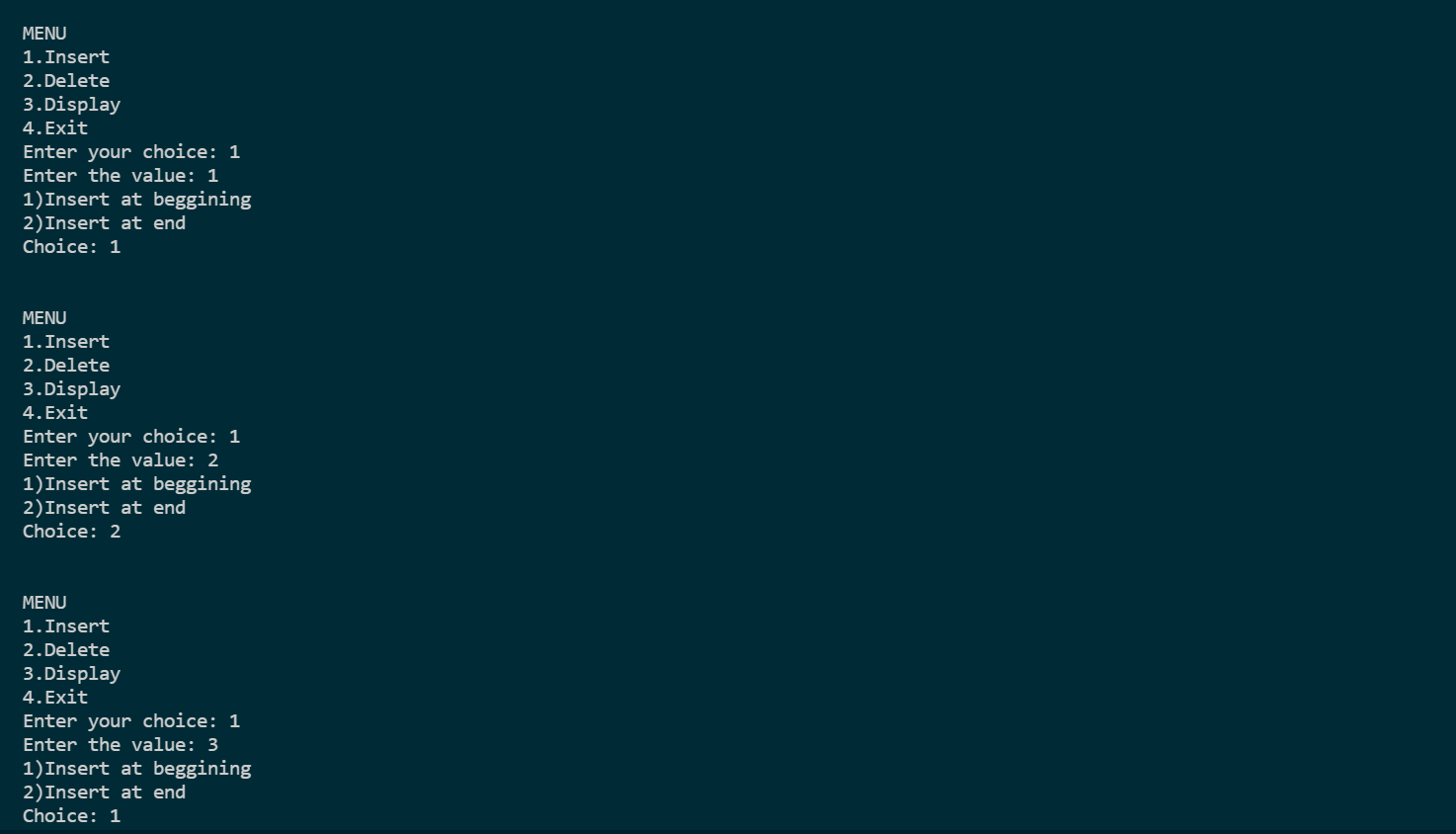
default:printf("\nINVALID INPUT");

}

}

}

**OUTPUT**



Q) Program to represent Circular Linked List

**PROGRAM CODE**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

typedef struct node node;

node \*head=NULL;

node \*tail=NULL;

node \*temp=NULL;

void ins(){

int n,ch;

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

printf("Enter the value: ");

scanf("%d", &n);

temp->data = n;

printf("1)Insert at beggining\n2)Insert at end\nChoice: ");

scanf("%d", &ch);

if (head == NULL){

head = tail = temp;

tail->next = head;

}

else

{

if (ch == 1)

{

temp->next=head;

head=temp;

tail->next=head;

}

else if(ch==2)

{

tail->next=temp;

tail=temp;

tail->next=head;

}

else

printf("\nINVALID INPUT");

}

}

void del(){

int ch;

node \*p;

if(head==NULL)

printf("The list is empty");

else if(head==tail)

{

free(head);

head=tail=NULL;

}

else

{

printf("1)Delete from beggining\n2)Delete from end\nChoice: ");

scanf("%d", &ch);

if (ch == 1)

{

temp=head;

head=head->next;

tail->next=head;

free(temp);

}

else if(ch==2)

{

p = head;

while (p->next != tail)

p = p->next;

temp = p->next;

tail = p;

tail->next = head;

free(temp);

}

else

printf("INVALID INPUT");

}

}

void disp(){

temp=head;

if(head==NULL)

printf("List is empty");

else

{

printf("Displaying the list: ");

do

{

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

temp=temp->next;

}while (temp!= head);

}

}

void main()

{

int ch;

while (1)

{

printf("\n\nMENU\n1.Insert\n2.Delete\n3.Display\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 1:

ins();

break;

case 2:

del();

break;

case 3:

disp();

break;

case 4:

exit(0);

break;

default:

printf("\nINVALID INPUT");

}

}

}

**OUTPUT**

