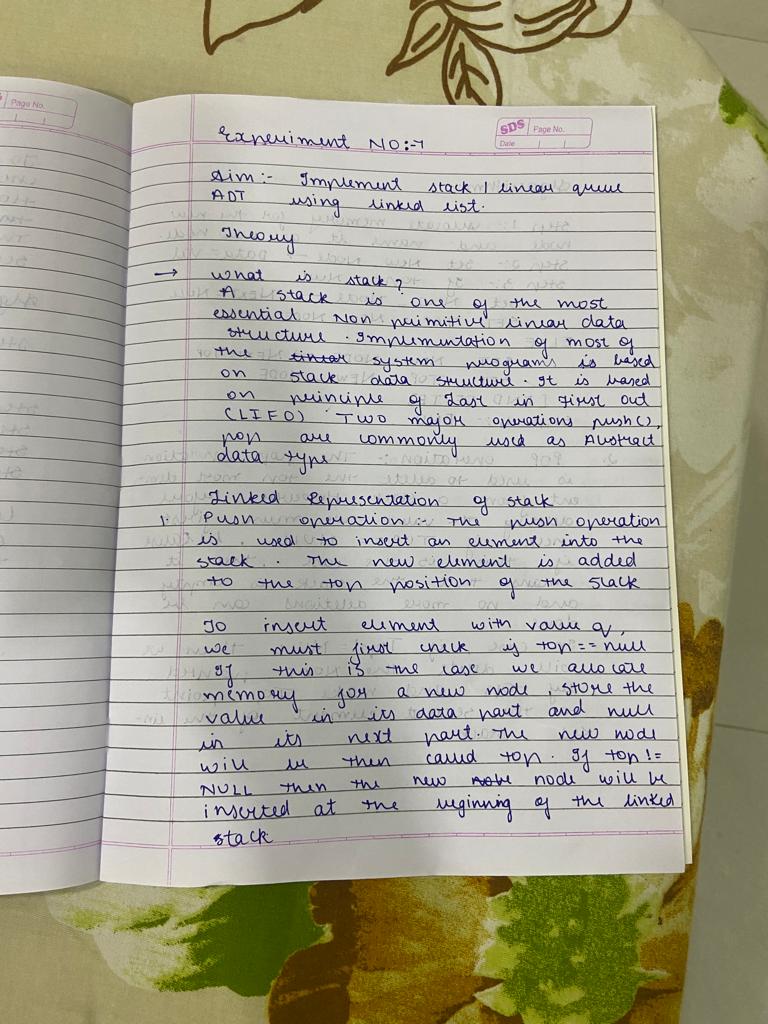
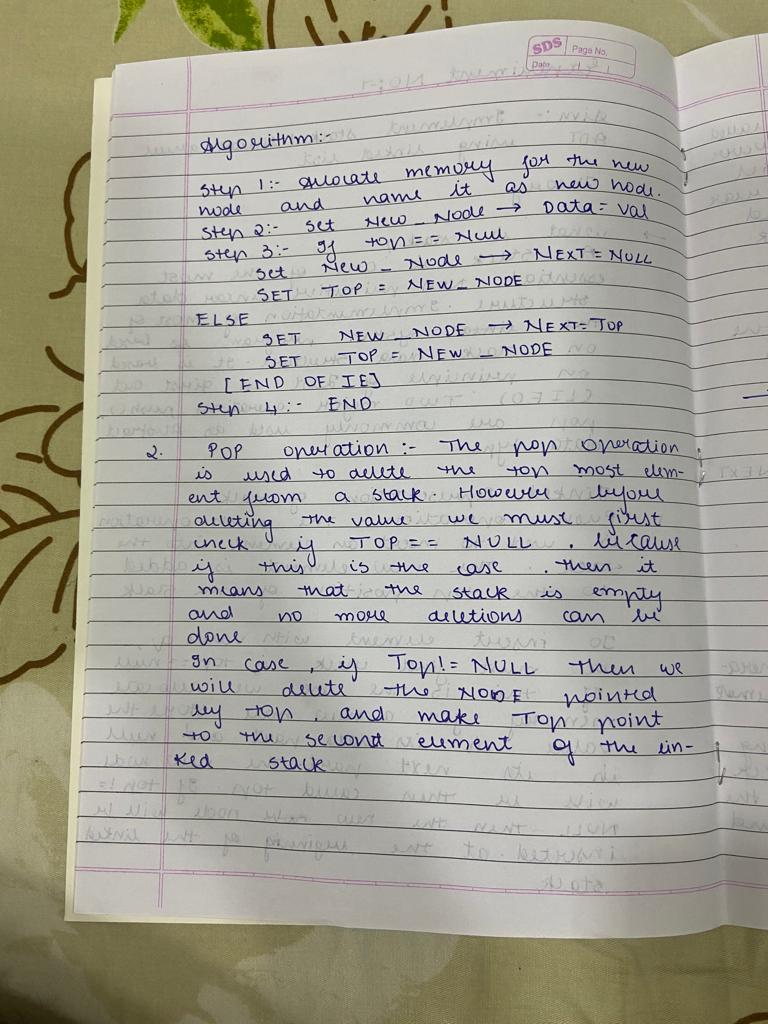
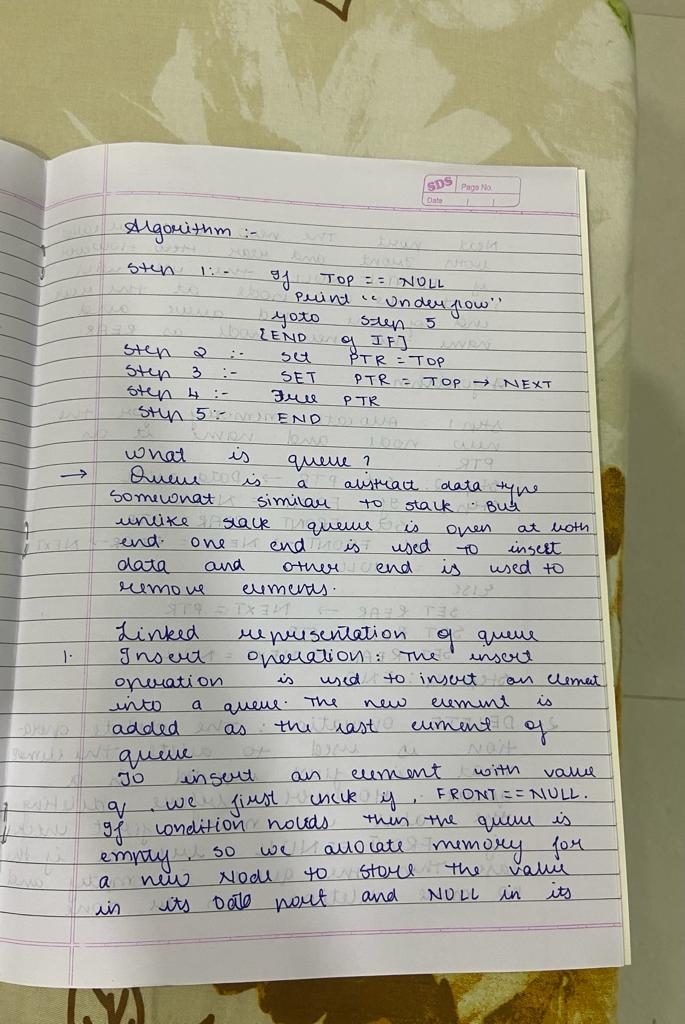


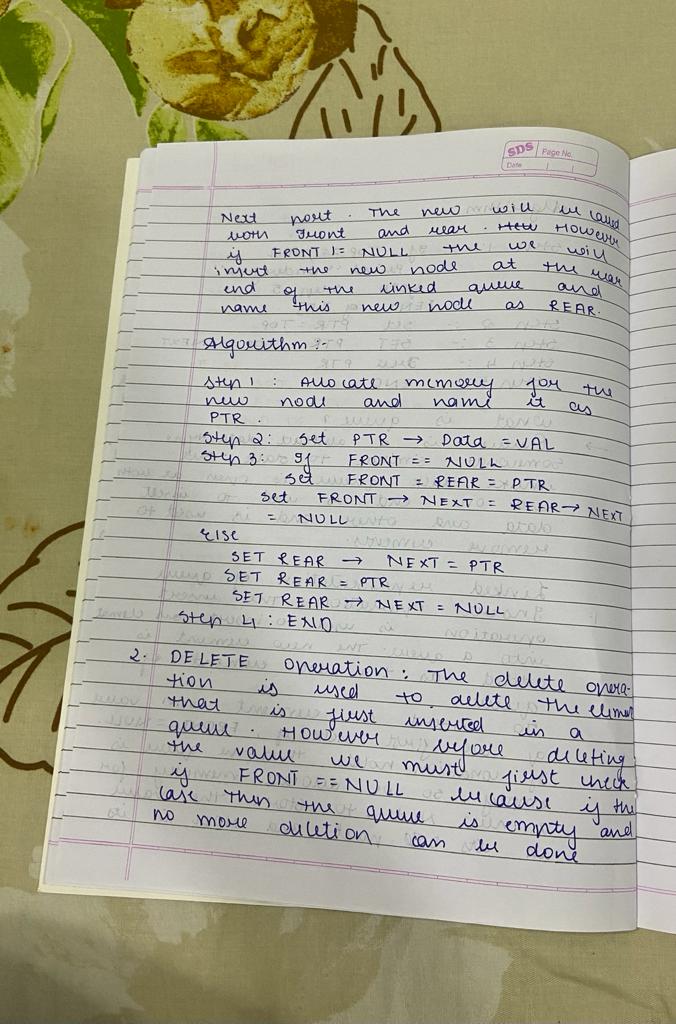
**COMPUTER ENGINEERING**

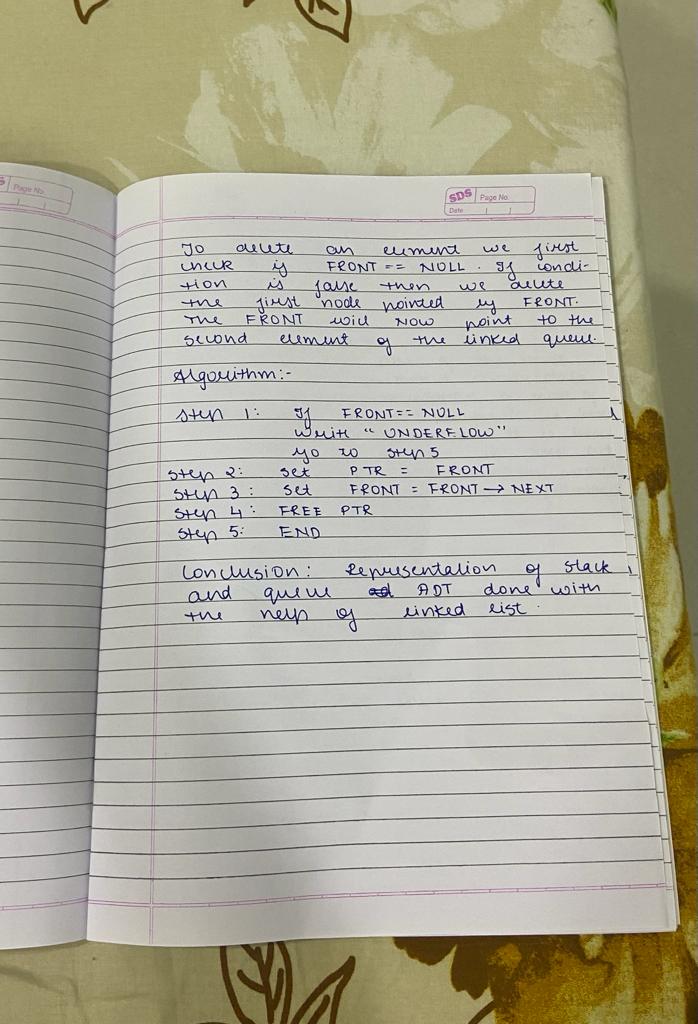
**DS ODD SEM 2021-22/EXPERIMENT 7 NAME:- GAURAV AMARNANI (D7A, 67)**



****

****

****

****

## PROGRAM NO 1: WRITE A PROGRAM TO IMPLEMENT STACK USING LINKED LIST

#include<stdio.h> #include<stdlib.h> #include<conio.h> #include<malloc.h> struct stack

{

int data;

struct stack \*next;

};

struct stack \*top = NULL;

struct stack \*push(struct stack \*, int); struct stack \*display(struct stack \*); struct stack \*pop(struct stack \*);

int peek(struct stack \*);

int main(int argc, char \*argv[]) { int val, option;

do

{

printf("\n \*\*\*\*\*MAIN MENU\*\*\*\*\*");

printf("\n 1. PUSH AN ELEMENT ONTO THE STACK"); printf("\n 2. POP AN ELEMENT FROM THE STACK");

printf("\n 3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK");

printf("\n 4. DISPLAY THE STACK");

printf("\n 5. EXIT");

printf("\n Enter your option: "); scanf("%d", &option); switch(option)

{

case 1:

printf("\n Enter the number to be pushed on stack: "); scanf("%d", &val);

top = push(top, val); break;

case 2:

top = pop(top); break;

case 3:

val = peek(top);

if (val != -1)

printf("\n The value at the top of stack is: %d", val); else

printf("\n STACK IS EMPTY"); break;

case 4:

top = display(top); break;

}

}while(option != 5);

return 0;

}

struct stack \*push(struct stack \*top, int val)

{

struct stack \*ptr;

ptr = (struct stack\*)malloc(sizeof(struct stack)); ptr -> data = val;

if(top == NULL)

{

ptr -> next = NULL; top = ptr;

}

else

{

ptr -> next = top; top = ptr;

}

return top;

}

struct stack \*display(struct stack \*top)

{

struct stack \*ptr; ptr = top;

if(top == NULL)

printf("\n STACK IS EMPTY"); else

{

while(ptr != NULL)

{

printf("\n %d", ptr -> data); ptr = ptr -> next;

}

}

return top;

}

struct stack \*pop(struct stack \*top)

{

struct stack \*ptr; ptr = top;

if(top == NULL)

printf("\n STACK UNDERFLOW");

else

{

top = top -> next;

printf("\n The value being deleted is: %d", ptr -> data); free(ptr);

}

return top;

}

int peek(struct stack \*top)

{

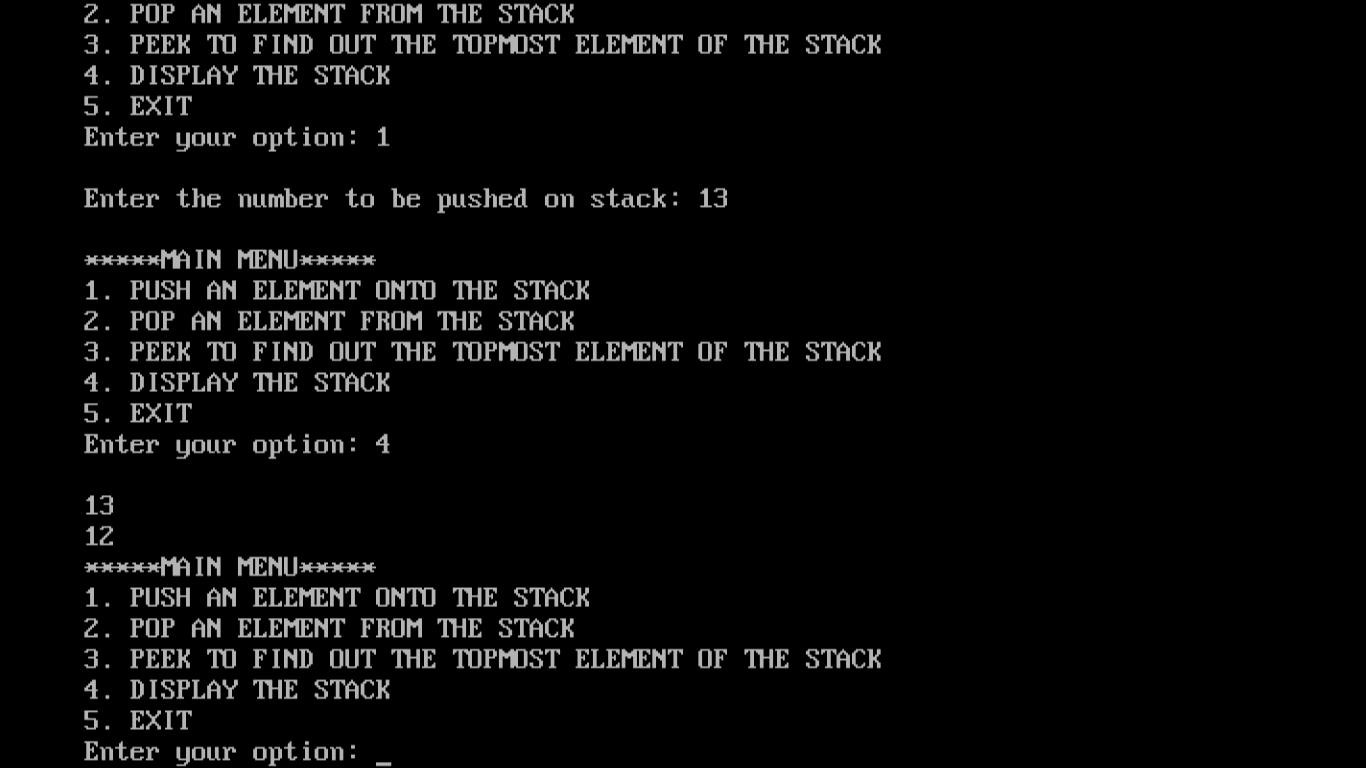
if(top==NULL) return -1;

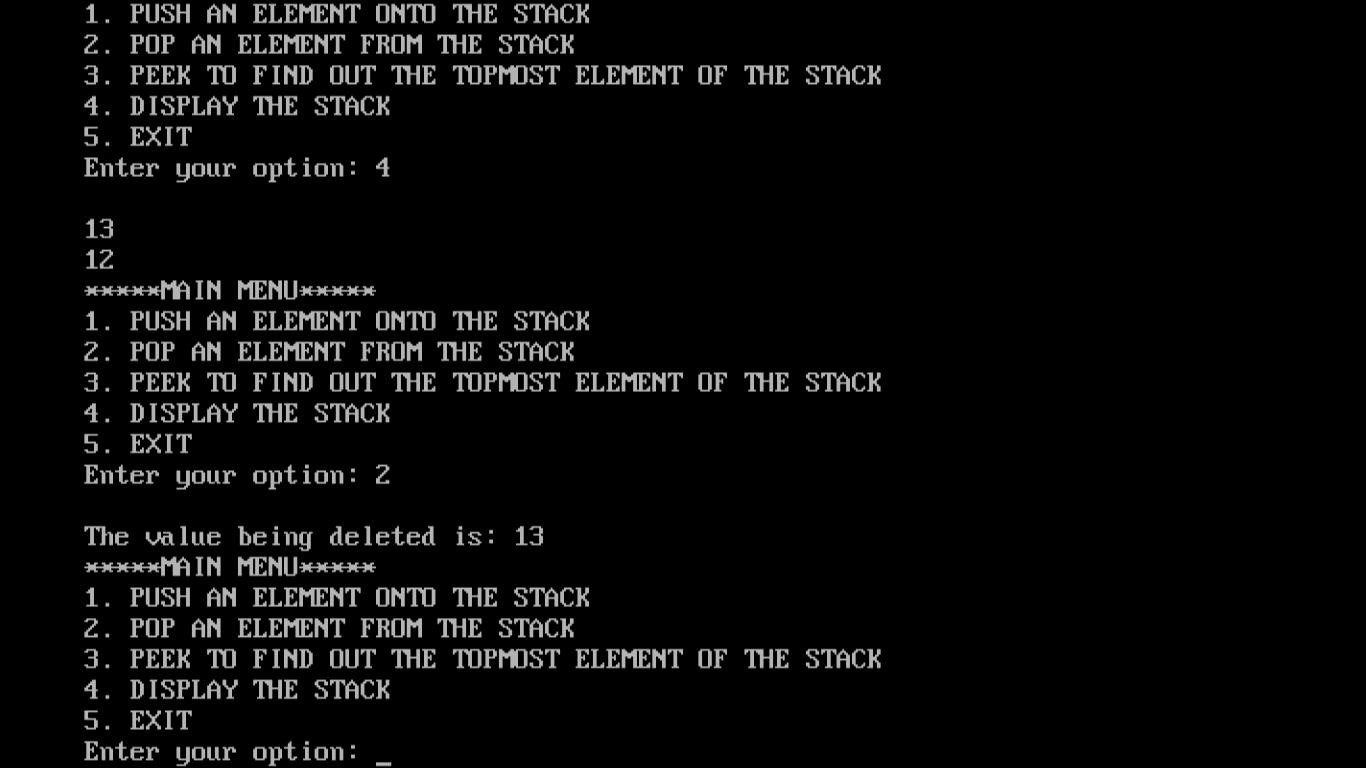
else

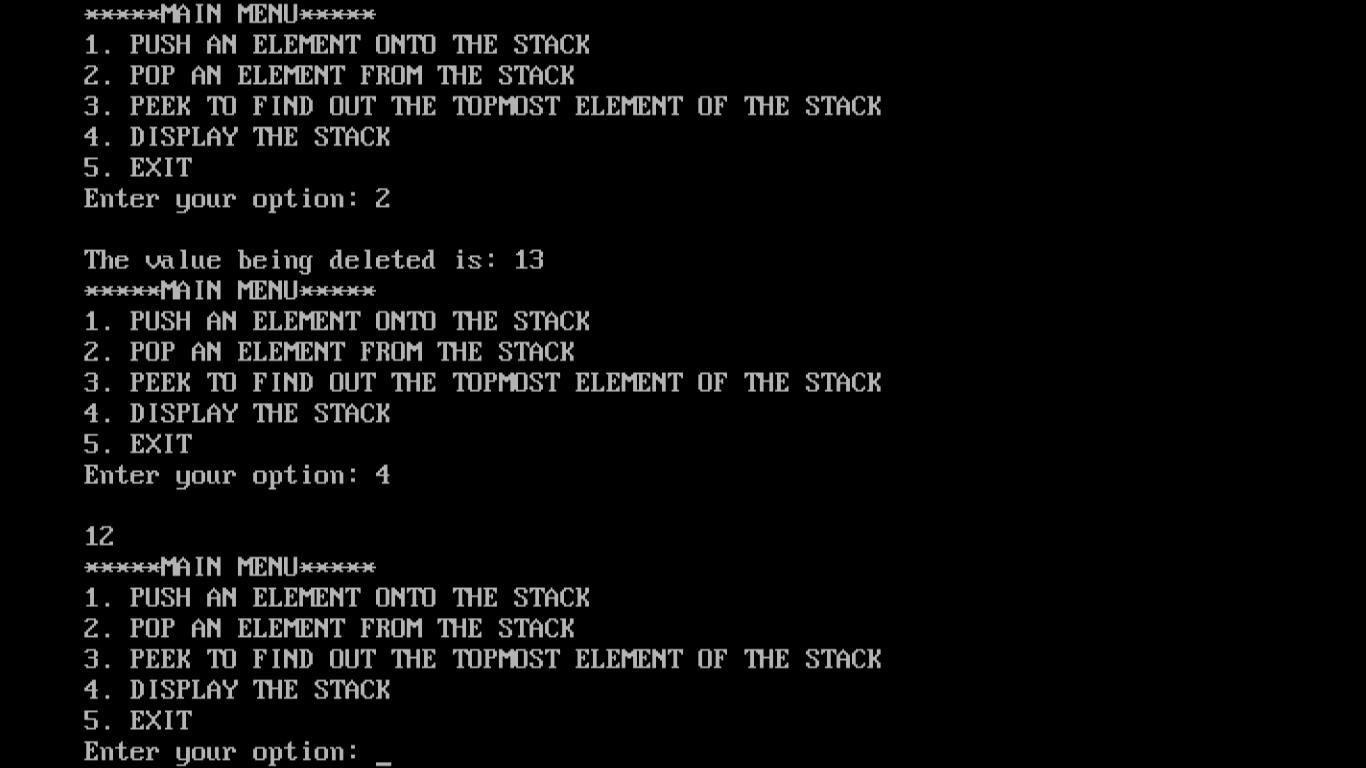
return top ->data;

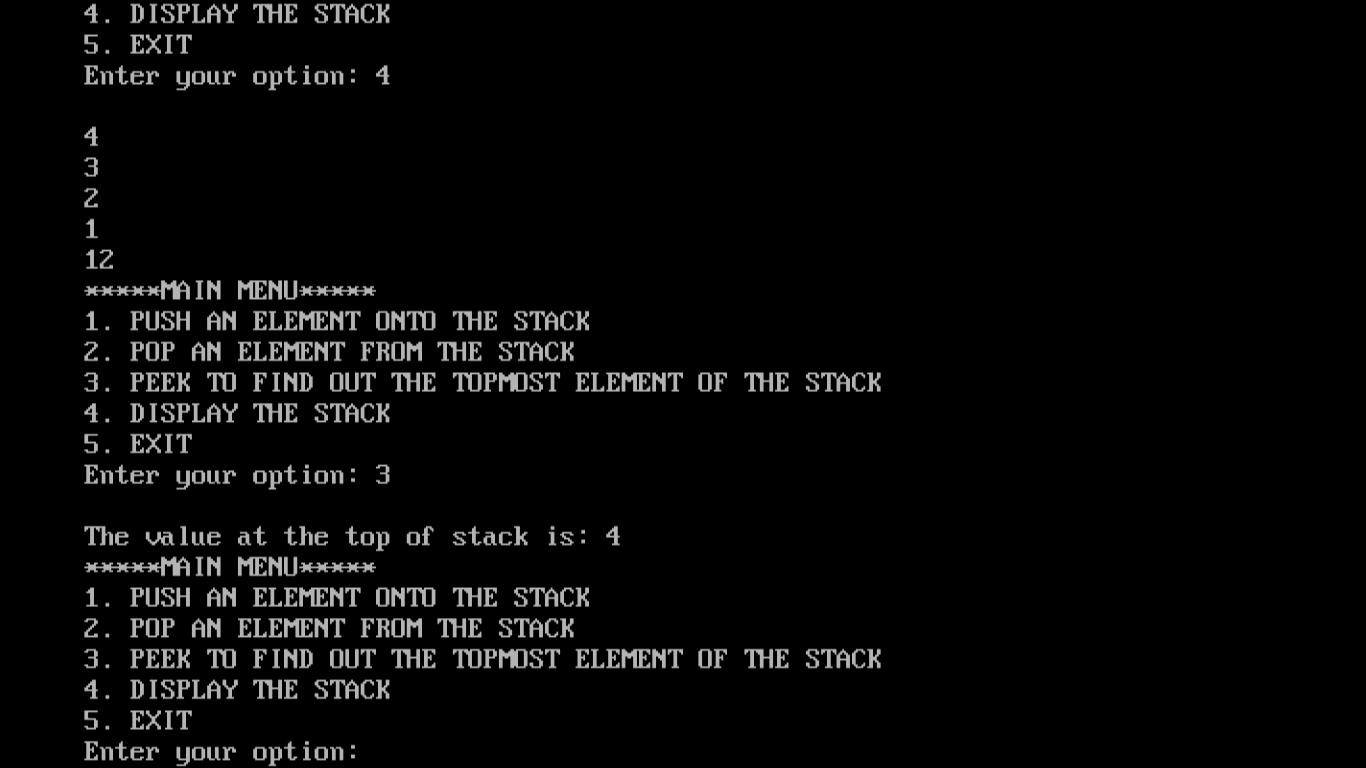
}

## OUTPUT:-









**PROGRAM NO 2: WRITE A PROGRAM TO IMPLEMENT QUEUE USING LINKED LIST**

#include <stdio.h> #include <conio.h> #include <malloc.h> struct node

{

int data;

struct node \*next;

};

struct queue

{

struct node \*front; struct node \*rear;

};

struct queue \*q;

void create\_queue(struct queue \*); struct queue \*insert(struct queue \*,int);

struct queue \*delete\_element(struct queue \*); struct queue \*display(struct queue \*);

int peek(struct queue \*); int main()

{

int val, option; create\_queue(q);

clrscr(); do

{

printf("\n \*\*\*\*\*MAIN MENU\*\*\*\*\*"); printf("\n 1. INSERT");

printf("\n 2. DELETE");

printf("\n 3. PEEK");

printf("\n 4. DISPLAY");

printf("\n 5. EXIT");

printf("\n Enter your option : "); scanf("%d", &option); switch(option)

{

case 1:

printf("\n Enter the number to insert in the queue:"); scanf("%d", &val);

q = insert(q,val); break;

case 2:

q = delete\_element(q); break;

case 3:

val = peek(q); if(val != -1)

printf("\n The value at front of queue is : %d", val); break;

case 4:

q = display(q); break;

}

}while(option != 5); getch();

return 0;

}

void create\_queue(struct queue \*q)

{

q -> rear = NULL; q -> front = NULL;

}

struct queue \*insert(struct queue \*q,int val)

{

struct node \*ptr;

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

ptr -> data = val;

if(q -> front == NULL)

{

q -> front = ptr; q -> rear = ptr;

q -> front -> next = q -> rear -> next = NULL;

}

else

{

q -> rear -> next = ptr; q -> rear = ptr;

q -> rear -> next = NULL;

}

return q;

}

struct queue \*display(struct queue \*q)

{

struct node \*ptr; ptr = q -> front; if(ptr == NULL)

printf("\n QUEUE IS EMPTY"); else

{

printf("\n"); while(ptr!=q -> rear)

{

printf("%d\t", ptr -> data); ptr = ptr -> next;

}

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

}

return q;

}

struct queue \*delete\_element(struct queue \*q)

{

struct node \*ptr; ptr = q -> front;

if(q -> front == NULL) printf("\n UNDERFLOW"); else

{

q -> front = q -> front -> next;

printf("\n The value being deleted is : %d", ptr -> data); free(ptr);

}

return q;

}

int peek(struct queue \*q)

{

if(q->front==NULL)

{

printf("\n QUEUE IS EMPTY"); return -1;

}

else

return q->front->data;

}

## OUTPUT:-

