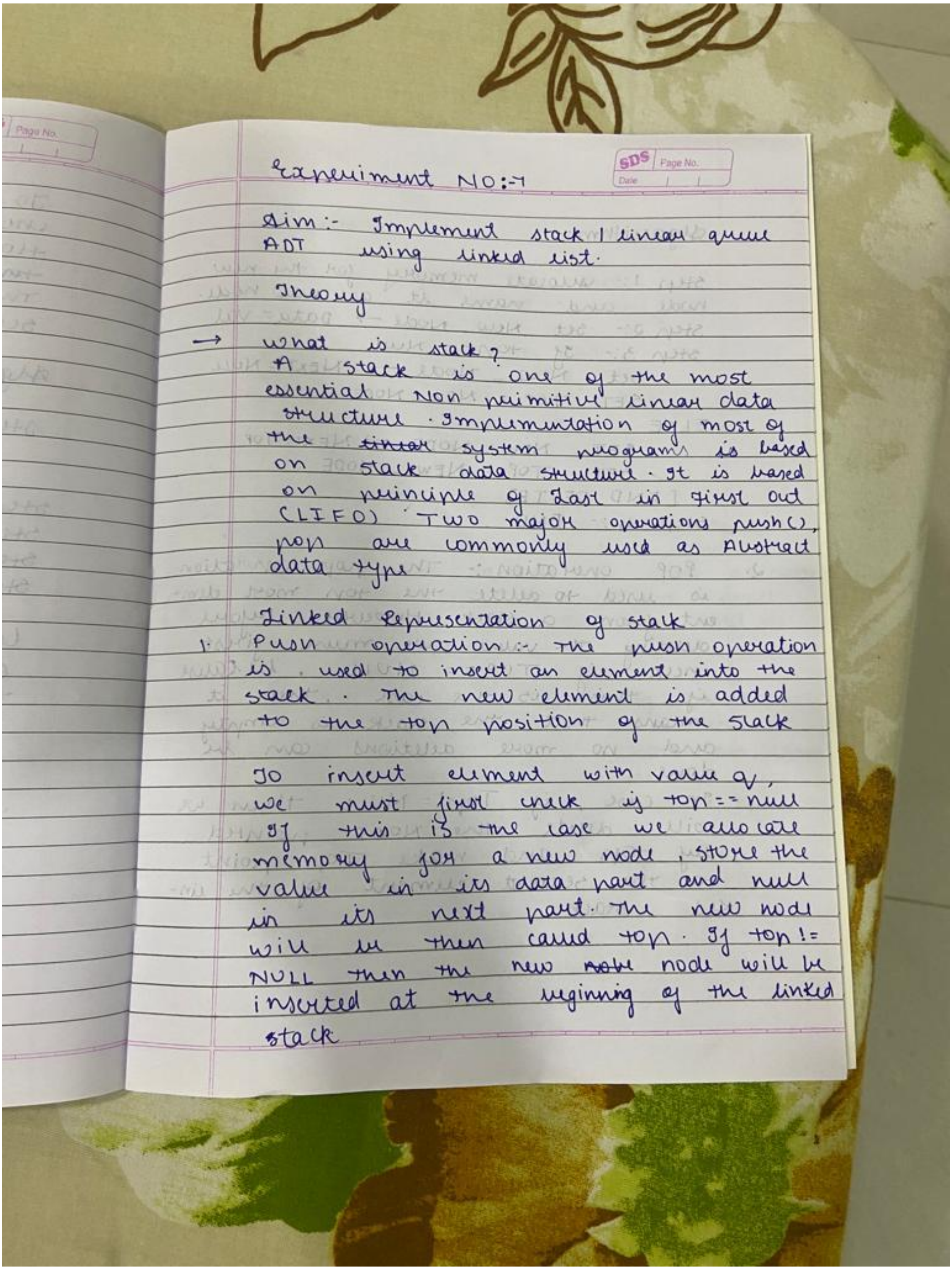
Vivekanand Education Society's Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

COMPUTER ENGINEERING

DS ODD SEM 2021-22/EXPERIMENT 7

NAME:- GAURAV AMARNANI (D7A, 67)



Experiment NO:-7

Aim :- Implement stack / linear queue ADT using linked list.

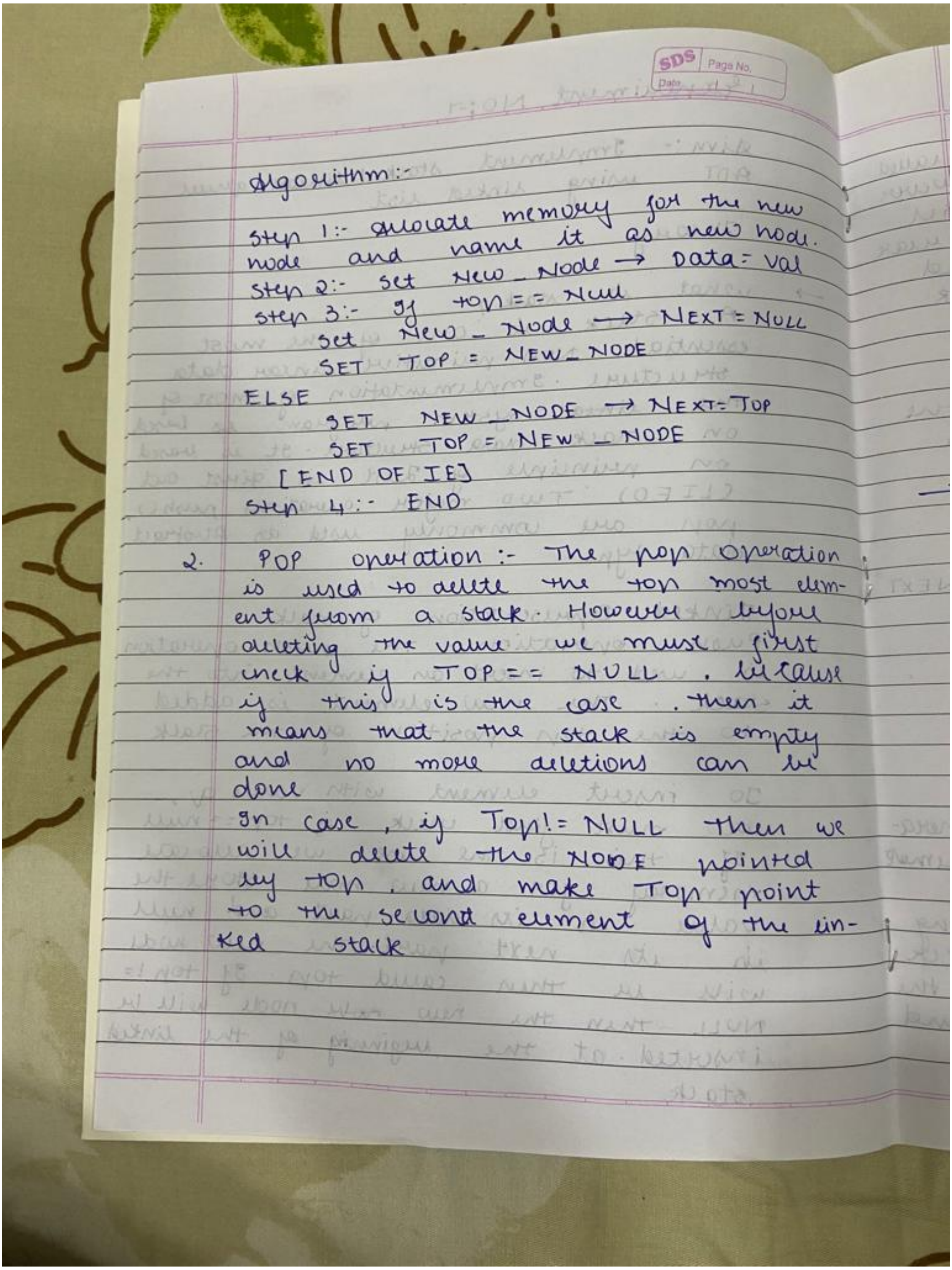
Theory

→ what is stack?
A stack is one of the most essential Non primitive linear data structure. Implementation of most of the ~~linear~~ system programs is based on stack data structure. It is based on principle of Last in First out (LIFO). Two major operations push(), pop are commonly used as Abstract data type

Linked Representation of stack

1. Push operation :- The push operation is used to insert an element into the stack. The new element is added to the top position of the stack

To insert element with value a, we must first check if top == null. If this is the case we allocate memory for a new node, store the value in its data part and null in its next part. The new node will be then called top. If top != NULL then the new ~~node~~ node will be inserted at the beginning of the linked stack



Algorithm:-

```
Step 1:- allocate memory for the new node and name it as new node.
Step 2:- set New_Node → Data = val
Step 3:- If top == Null
    set New_Node → NEXT = NULL
    SET TOP = NEW_NODE
ELSE
    SET NEW_NODE → NEXT=TOP
    SET TOP = NEW_NODE
  [END OF IE]
Step 4:- END
```

2. POP operation :- The pop operation is used to delete the top most element from a stack. However before deleting the value we must first check if TOP == NULL. because if this is the case. then it means that the stack is empty and no more deletions can be done

In case, if Top!= NULL then we will delete the NODE pointed by top, and make Top point to the second element of the linked stack

Algorithm :-

```
Step 1 :-  If TOP == NULL
               Print "Underflow"
               goto step 5
           [END of IF]
Step 2 :-  Set PTR = TOP
Step 3 :-  SET PTR = TOP → NEXT
Step 4 :-  Free PTR
Step 5 :-  END
```

What is queue?

→ Queue is a abstract data type somewhat similar to stack. But unlike stack queue is open at both end. one end is used to insert data and other end is used to remove elements.

Linked representation of queue

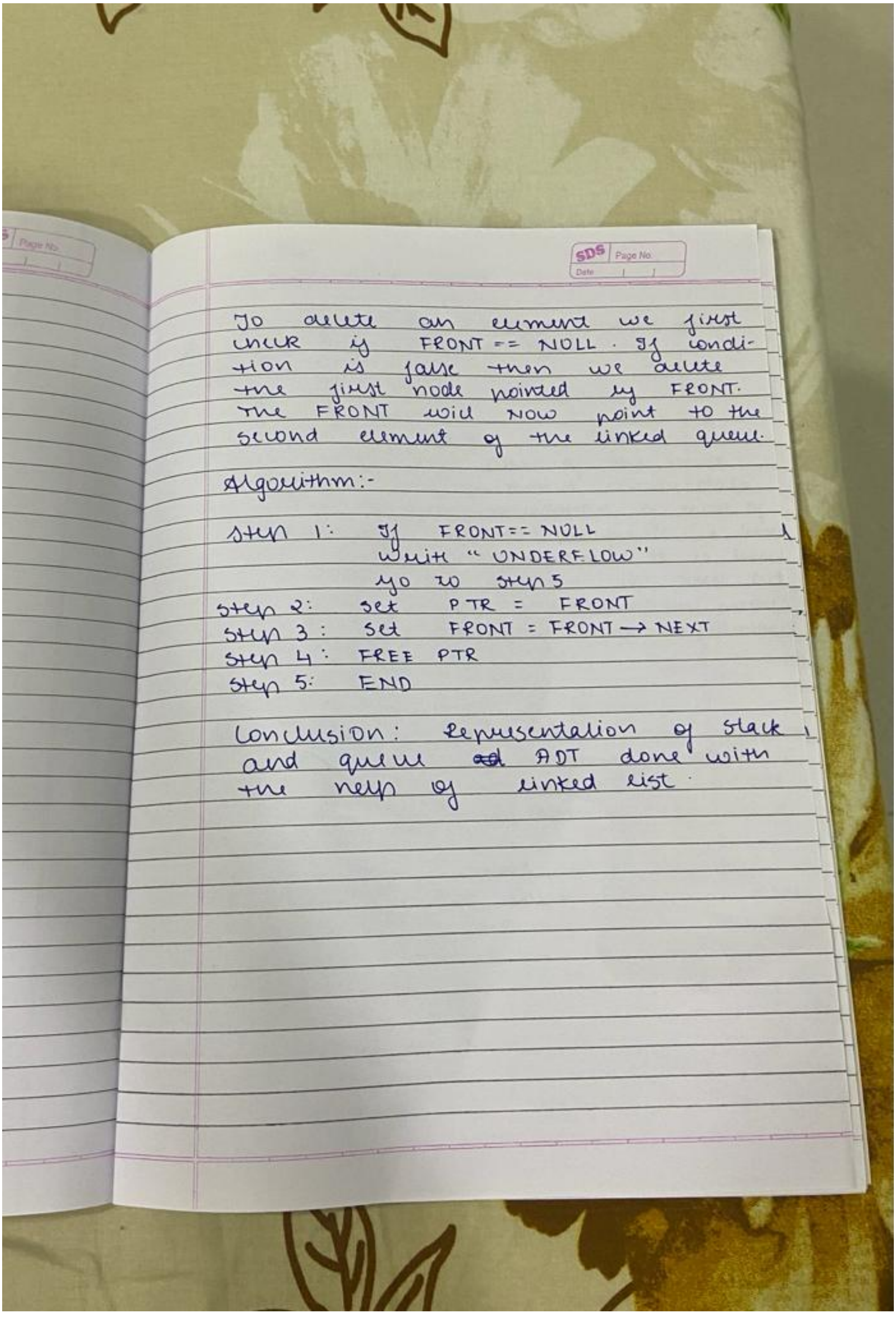
1. Insert operation: The insert operation is used to insert an element into a queue. The new element is added as the last element of queue
To insert an element with value 9, we first check if, FRONT == NULL. If condition holds then the queue is empty. so we allocate memory for a new Node to store the value in its Data part and NULL in its

Next part. The new will be called both front and rear. ~~Hew~~ However if FRONT != NULL the we will insert the new node at the rear end of the linked queue and name this new node as REAR.

Algorithm :-

```
Step 1 : Allocate memory for the new node and name it as PTR.
Step 2: Set PTR → Data = VAL
Step 3: If FRONT == NULL
          Set FRONT = REAR = PTR
        Set FRONT → NEXT = REAR → NEXT
          = NULL
   Else
        SET REAR → NEXT = PTR
        SET REAR = PTR
        SET REAR → NEXT = NULL
Step 4 : END
```

2. DELETE Operation : The delete operation is used to delete the element that is first inserted in a queue. However before deleting the value we must first check if FRONT == NULL because if the case then the queue is empty and no more deletion can be done



To delete an element we first check if FRONT == NULL. If condition is false then we delete the first node pointed by FRONT. The FRONT will now point to the second element of the linked queue.

Algorithm:-

```
Step 1:  If FRONT == NULL
             Write "UNDERFLOW"
             go to step 5
Step 2:  set PTR = FRONT
Step 3:  Set FRONT = FRONT → NEXT
Step 4:  FREE PTR
Step 5:  END
```

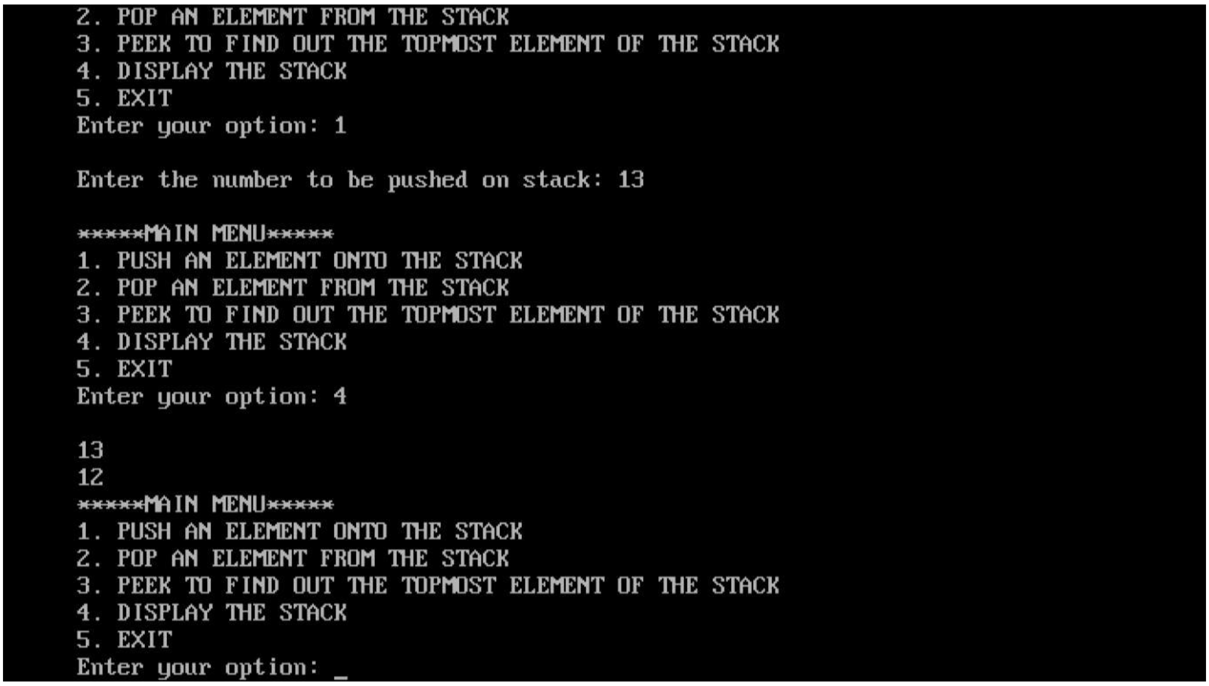
Conclusion: Representation of stack and queue ~~ad~~ ADT done with the help of linked list.

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

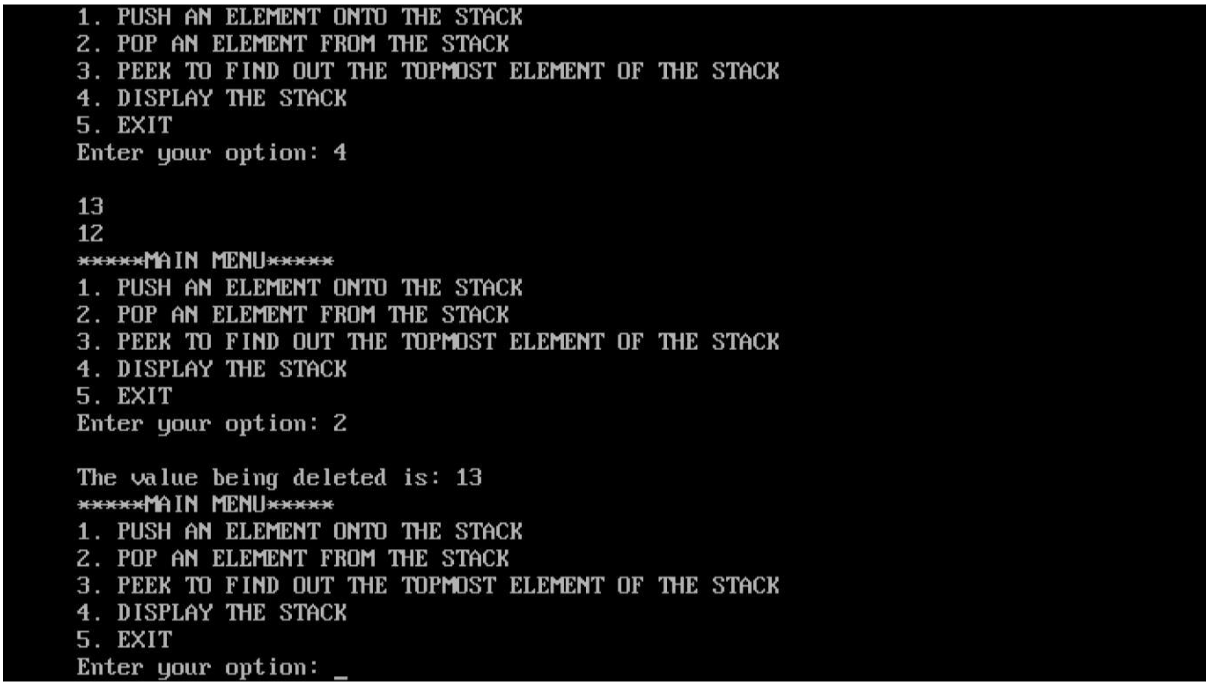
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<malloc.h>
struct stack
{
int data;
struct stack *next;
};
struct stack *top = NULL;
struct stack *push(struct stack *, int);
struct stack *display(struct stack *);
struct stack *pop(struct stack *);
int peek(struct stack *);
int main(int 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;
}
```

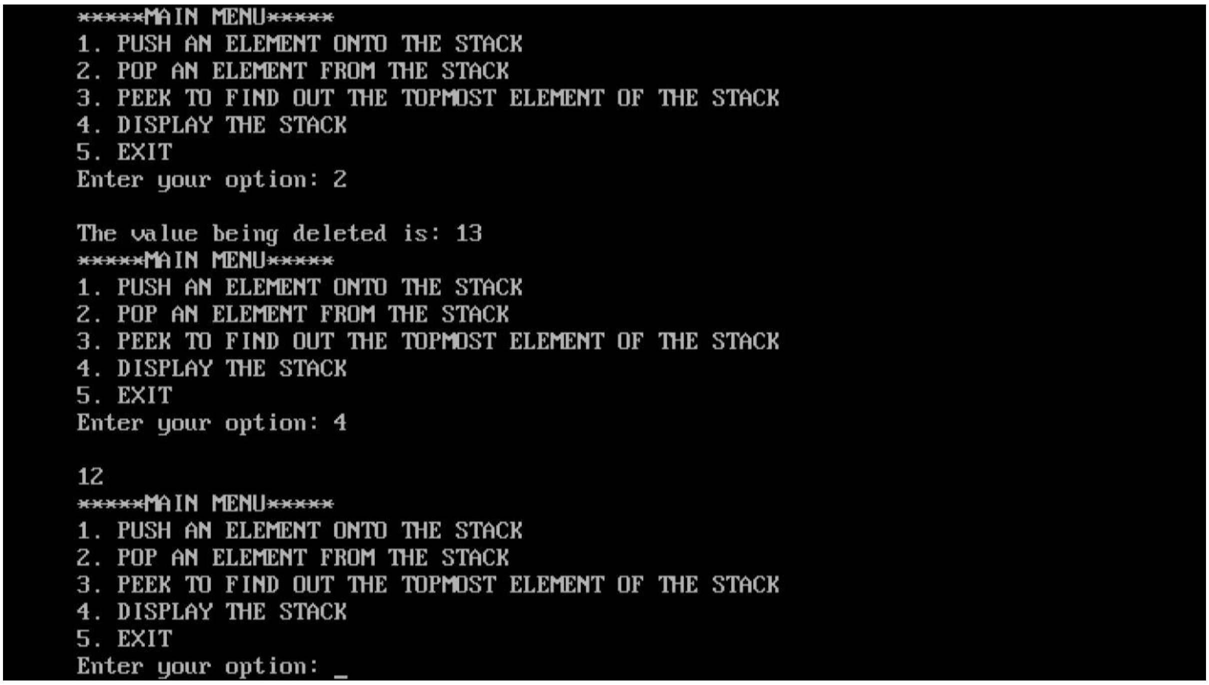
OUTPUT:-



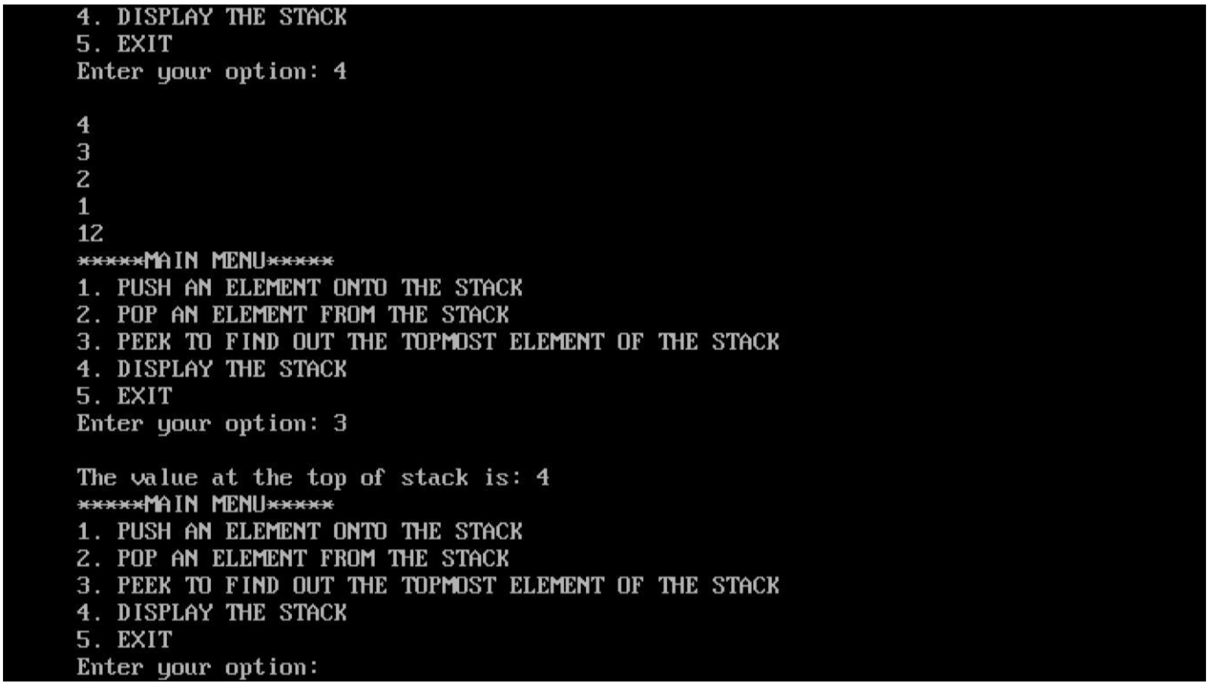
```
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 1

Enter the number to be pushed on stack: 13

*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 4

13
12
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: _
```



```
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 4

13
12
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 2

The value being deleted is: 13
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: _
```



```
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 2

The value being deleted is: 13
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 4

12
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: _
```



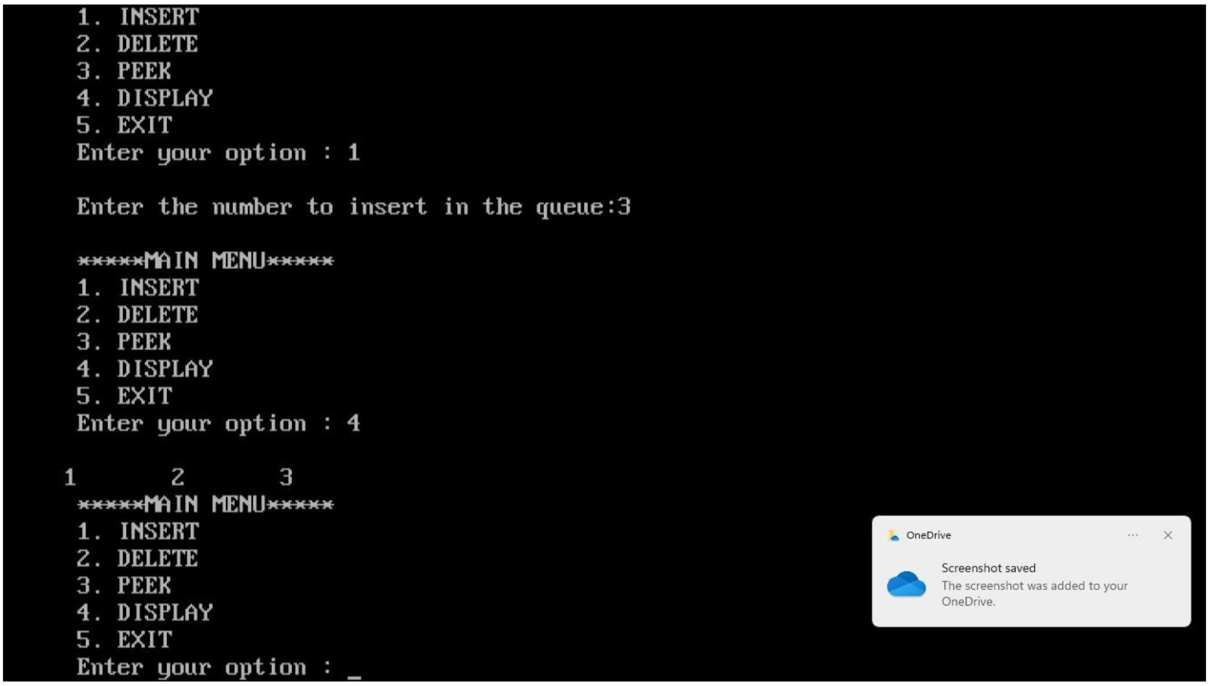
```
4. DISPLAY THE STACK
5. EXIT
Enter your option: 4

4
3
2
1
12
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option: 3

The value at the top of stack is: 4
*****MAIN MENU*****
1. PUSH AN ELEMENT ONTO THE STACK
2. POP AN ELEMENT FROM THE STACK
3. PEEK TO FIND OUT THE TOPMOST ELEMENT OF THE STACK
4. DISPLAY THE STACK
5. EXIT
Enter your option:
```

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:-

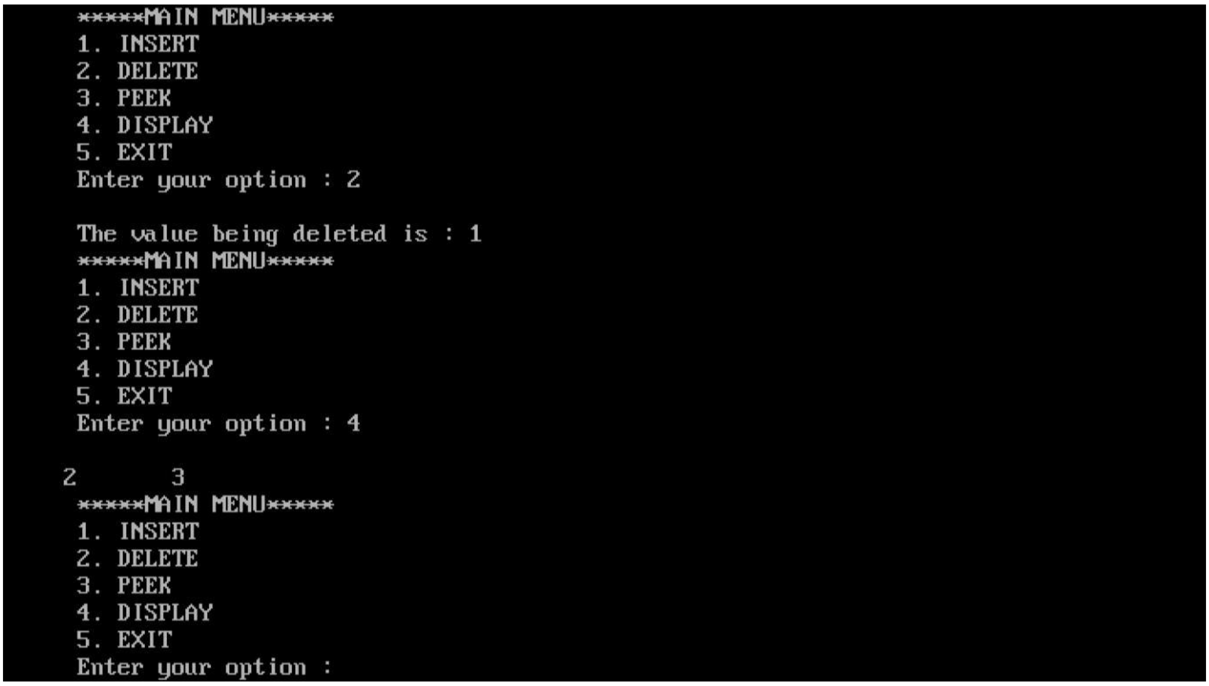


```
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : 1

 Enter the number to insert in the queue:3

 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : 4

1       2       3
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : _
```

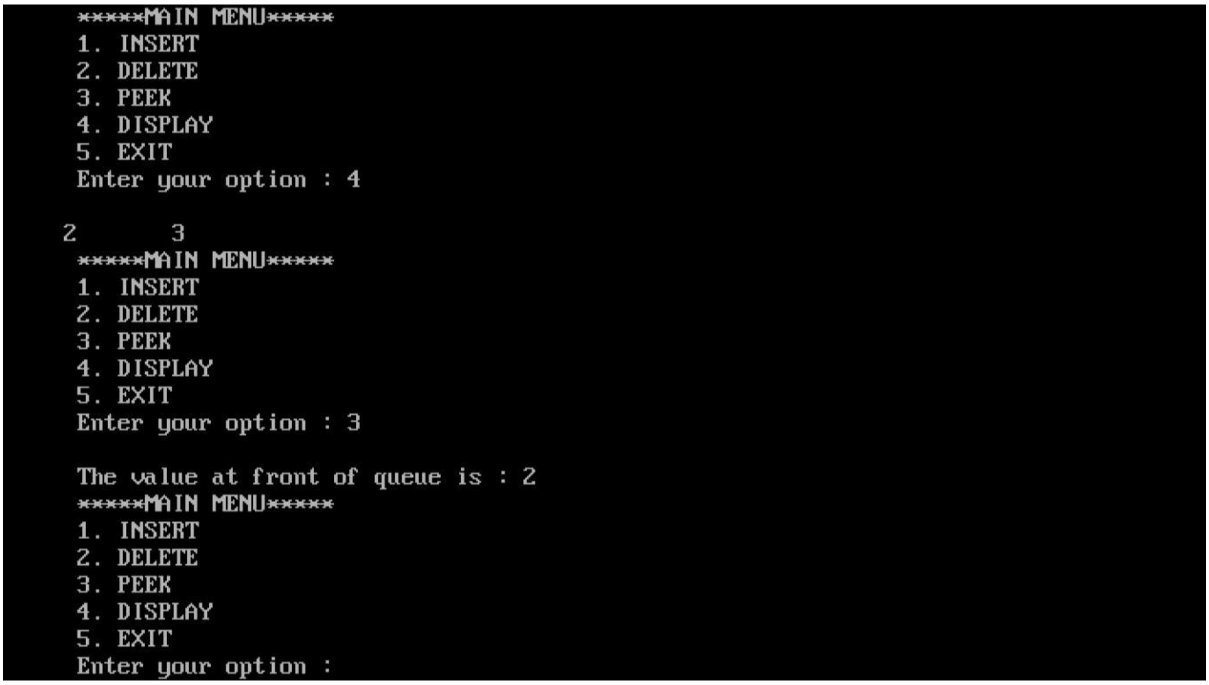
OneDrive

Screenshot saved

The screenshot was added to your OneDrive.



```
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : 2

 The value being deleted is : 1
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : 4

2       3
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option :
```



```
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : 4

2       3
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option : 3

 The value at front of queue is : 2
 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. PEEK
 4. DISPLAY
 5. EXIT
 Enter your option :
```