Register No: 20L31A0464 Experiment No: 4 Date: 20|4|22

S. No	Component	Max. Marks	Marks Secured
1	Preparedness	2	2
2	Viva-Voce	2	2
3	Experiment	3	3
4	Analysis & Record	3	3
	Total	10	10
Date	20|4|22	Signature of the Lab teacher	

AIM: Write a program to implement the stack using arrays and linked list.

Description :

stack: A stack is a linear data structure. Consisting of a set of elements and is based on principle last in first out (LIFO)

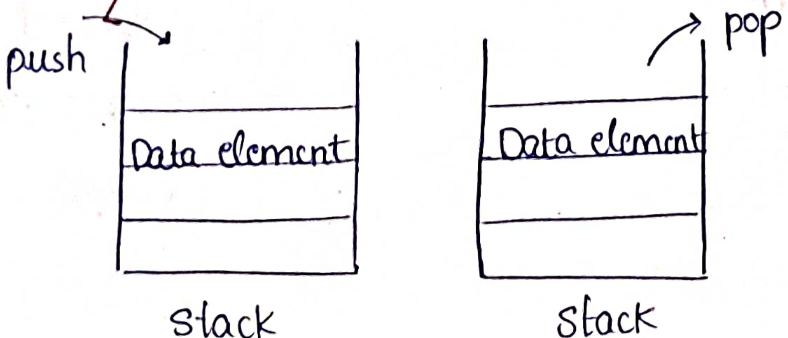
→ LIFO: The element which is entered at last will be coming first.

→ In stack, adding and removing an element at same end is called top of a stack.

→ The two basic operations associated with stack are

push: is term used to insert element into stack.

pop: is term used to delete element from stack.



Output :

```
 * * * * MENU * * * *
 1. Push
 2. Pop
 3. Display
 4. Exit
 Enter your choice : 3
 Stack is empty
 * * * MENU * * *
 1. Push
 2. Pop
 3. Display
 4. Exit
 Enter your choice :1
Enter value to be sorted : 10
 Insertion successful!!
 * * * MENU * * *
1. Push
2. Pop
3. Display
4 Exit
Enter your choice 3
Stack elements are 10.
```

Register No : 64 Experiment No : 4 Date: 20/4/22

Program: (arrays)

```
#include<stdio.h>
#include<conio.h>
#define size10
void push(int);
 void pop();
 void display();
 int stack[stack],top=-1;
void main()
{
int value,choice;
clrscr();
while(1)
{
printf("\n\n**** MENU****\n");
printf("1.push \n 2. pop \n 3. Display \n 4. Exit \n");
Scanf("%d", &choice);
Switch (choice)
{
Case 1: Print f("Enter the value to be insert: ");
        Scanf("%d", &value);
        push(value);
        break;
Case 2: pop();
        break;
```

Register No : 64 Experiment No : 4 Date: 20/4/22

```
Case 3: display();
        break;
Case 4: Exit(0);
default: print f(" \n wrong selection))) try again !!);
}
}
void push (int value)
{
if (top == size-1)
print f("\n stack is full )!, insertion is not possible!!!");
  else
  {
  top ++;
  stack [top] = value;
  print f ("\n insertion success!!!");
  }}
  void pop () {
   if (top == -1)
  print f ("\n stack is empty!!! Deletion is not possible!!!");
  else
  {
  printf ("\n deleted: %d", stack[top]);
  else {
  int i;
  print f(" \n stack elements are : \n");
  for (i=top ; i>=0 ; i--)
  printf(" %d "\n, stack[i]);
}
}
}
```

Output :

```
Stack using linked list ::
1. Push
2. Pop
3. display .
4. Exit
Enter your choice :1
Enter the data to push :10
Insertion is success !!!
Enter your choice : 3
10 ------> NULL
Enter your choice : 1
  Enter the data to push : 20
  Insertion is Success !!!
Enter your choice 2
    deleted element : 20
Enter your choice : 3
10 ---> NULL
```

Register No : 64 Experiment No : 4 Date: 20/4/22

Program:

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
struct node
{
int data;
struct node *next;
}
int data;
 struct node *next;
}
*top ; *new ; *temp ;
void push();
void pop();
 void display();
void main()
 {
 int choice ; value;
print f("\n: stack using linked list ::\n");
while(1){
print f("1. push\n 2. pop \n 3. display \n 4. exit \n");
print f("enter your choice");
 Scanf(" %d", choice);
 Switch(choice){
  Case 1: push();
     break;
```

Register No : 64 Experiment No : 4 Date: 20/4/22

```
Case 2: pop()
   break;
Case 3: display (): break .
Case 4: exit(0);
default: printf(" \n wrong selection !!) please
}}}
void push ()
{
int x;
printf("enter the data to be push");
Scanf (" %d", &x);
new = (struct node * ) malloc (size of struct node));
 new → data =value;
 if (top == NULL)
 new → next = NULL;
else
new→next = Top;
  top = new;
 printf("\n Insertion done \n");
}
 void pop ( )
{
if (toop= = NULL)
printf(" \n stack is empty !!! \n");
else
{
temp = top;
print f("\n deleted element %d", temp → data
 top = temp → next ;
 free (temp);
 }}
void display ( )
{
if (top == NULL)
```

```
    print f (" /n stack is emply !!! \n");
  else {
  temp = top;
  while (temp != NULL)
 {
 print f(" %d → ", temp → data);
 temp = temp → next ;
 }}}
```

b) write a program to implement queue using array and linked list.

Aim : To write a program to implement Queue using arrays and linked list.

Description :

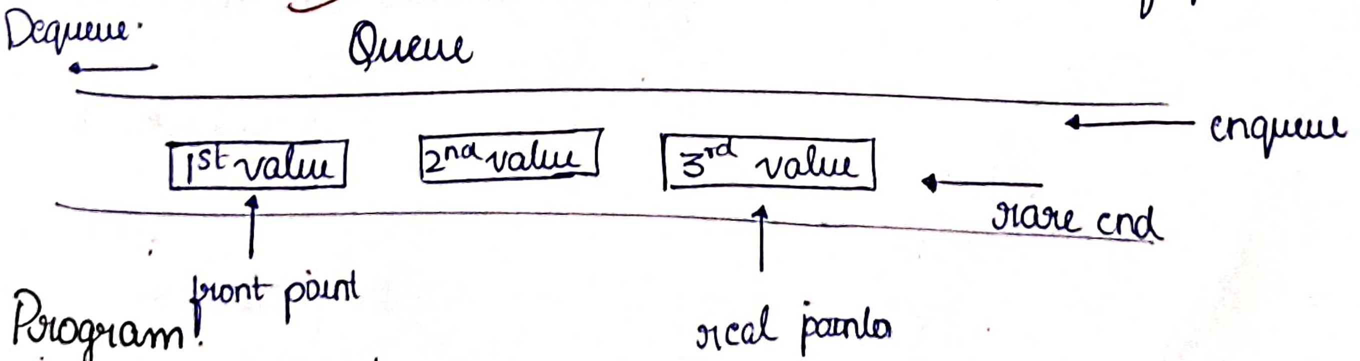
Queue :

→ A Queue is a linear data structure consisting of a set of element and is based on principle of first in first out.

→ FIFO : The elements which is executed at first will be coming first.

→ The basic two operations associated with Queue are

1. Enqueue which is sorts an element at end of queue.
2. Dequeue which deletes an element at start of queue.



Program!

```
#include <stdio.h>
#include define size 5
int q[SIZE];
int f=-1,r=-1;
```

Register No : 64 Experiment No : 4 Date: 20|4|22

```
int enqueue()
{
int x;
if(r==size-1)
{
 prant f("overflow");
}
else
{
print f(" enter  x value ");
Scanf("%d", &x);
 r = r+1
 q[r] = x;
}}
Void dequeue()
{
 if (f==-1&& r==-1)
{
 print f("\n Queue  is  empty \n");
{
else if(f==r)
 { printf(" \n The deleted element = %d \n", q[f]);
  f=-1;
  r=-1;
 }
 else
 { f=f+1;
 printf("\n the deleted element = %d \n", q[f]);
 }}
 Void display()
 {
 int i;
 if(f=-1 && r==-1)
 {
 print f("\n Queue is empty \n");
}
else
{
print f("\n The elements in Queue are \n");
```

Register No: 64 Experiment No: 4 Date: 20/4/22

S. No	Component	Max. Marks	Marks Secured
1	Preparedness	2	2
2	Viva-Voce	2	2
3	Experiment	3	3
4	Analysis & Record	3	3
	Total	10	10
Date		Signature of the Lab teacher	

```
      i = f+1;
      while (i<=r)
AIM: {
      printf("%d", q[i]);
       L = i+1
    }
  printf("%d", q[i]);
 }}
 void main ()
 {
 int ch, item;
  do
 {
 printf("\n 1. insert \n");
 print f(" \n 2. delete \n");
 print f(" \n 3. Display \n");
 print f("\n 4. Exit\n");
 printf ("\n Enter your choice \n");
 Scanf(" %d", &ch);
Switch (ch)
{
 Case 1: print f(" \n Enter elements to be pushed \n");
  Scanf ("%d", & item );
  add (item );
 break ;
Case 3: display( )
   break;
Case 4: Exit(0);
}
} while (ch! =4);
}

# include <stdio.h>
# include <conio.h>
Struct queue
{
int queue
{
int data;
Struct queue* next;
} * new node, * rear, * frong, * temp;
void enque ()
{
int x;
new node(Struct queue*) malloc (sizeof (Struct Queue));
printf("\n Enter data");
scanf ("%d", &x);
new node → data=x;
new node → next =NULL;
if (front== NULL && real==NULL)
{front= new node;
real= new node;
else
 {
 rear → next= new node;
rear → new node.
}}
void dequeue ( )
```

```
{
 if(front = = NULL ** rear = = NULL)
{
 printf ("\n \n\t emply queue");
}
temp= front ;
printf("\n\n\t deleted element from queue is %d", temp->data);
front = front -> next
  free(temp);
}
void display() {
if(front== NULL && rear == NULL)
printf("\n\n\t emply queue");
else {
temp= temp -> next;
}}
void main()
{
char ch;
printf("\n\t  Queue operations using pointers");
 printf(" \n 1. Insert ");
 printf("\n 2. delete");
 printf("\n 3. display");
 printf("\n 4. Quit");
 while(1)
{
printf(" enter your choice");
 Scanf("%d", &ch);
 Switch (ch)
 {
  Case 1:
 enqueue();
  break;
Case 2:
dequeue ();
break:
Case 3:
display () break;
Case 4:
exit(1); break;
}
getch();
}
```