* Problem **A**

**Stack**

**AIM :-** To write a ‘c’ Program to perform operations on **Stack** of Push and Pop**.**

**Theory :-** A **Stack** is also a **Linear Data Structure** which Stores the data. The Insertion and Deletion Happen from only one end. It uses the concept of **FILO** or **LIFO.**

**Algorithms :-** Step 1:- Start.

Step 2:- Declare integer Global Variables stack[10] and top=-1.

Step 3:- Stop.

UDF Push ()

Step 1:- Start.

Step 2:- Declare integer variable x and n=sizeof(stack)/sizeof(stack[0]).

Step 3:- Read variable ‘x’.

Step 4:- Check **if** condition top==n-1, if it is true go to step 5 otherwise go to step 6.

Step 5:- Print Stack is Full.

Step 6:- Increment top++ and stack[top]=x.

Step 7:- Stop.

UDF Pop ()

Step 1:- Start.

Step 2:- Check **IF** condition top==-1, if it is true go to step 3 otherwise go to step 4.

Step 3:- Print Stack is Empty.

Step 4:- Declare “loc” variable and assign loc=stack[top] and decrement top--.

Step 5:- Print Popped element and display “loc”.

Step 6:- Stop.

Main()

Step 1:- Start.

Step 2:- Declare ‘i’ variable for loop.

Step 3:- Call Push operation 6 times.

Step 4:- Call Pop operation 4 times.

Step 5:- Print stack by loop.

Step 6:- Stop.

**Program :-**

#include<stdio.h>

int stack[10]; //Global Variables

int top=-1;

void push()

{

int x, n=sizeof(stack)/sizeof(stack[0]);

printf("Enter the Element in the Stack???\n");

scanf("%d",&x);

if(top==n-1)

printf("Stack is full!!!\n");

else

{

top++;

stack[top]=x;

}

}

void pop()

{

if(top==-1)

printf("Stack is Empty\n");

else

{

int loc=stack[top];

top--;

printf("Popped element is %d\n",loc);

}

}

int main()

{

int i;

push();

push();

push();

push();

push();

push();

pop();

pop();

pop();

pop();

printf("\n\nElements of Stack are....\n");

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

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

return 0;

}

**Output :-**

Enter the Element in the Stack???

1

Enter the Element in the Stack???

2

Enter the Element in the Stack???

3

Enter the Element in the Stack???

4

Enter the Element in the Stack???

5

Enter the Element in the Stack???

6

Popped element is 6

Popped element is 5

Popped element is 4

Popped element is 3

Elements of Stack are....

1

2

**Observation :-** After performing the experiment we observed that Values can be stored in a Stack and we were able to use UDF Push and Pop operations. Insertion and Deletion of Values happened from one end only. It took 0.31 sec Compilation Time.

* Problem **B**

**Queue**

**AIM :-** To write a ‘c’ Program to perform operations on **Queue** of **Enqueue** and **Dequeue.**

**Theory :-** A **Queue** is also a **Linear Data Structure** which Stores the data. The Insertion and Deletion Happen from only Two end. The Insertion happens from last and Deletion from First. It uses the concept of **FIFO.**

**Algorithms :-** Step 1:- Start.

Step 2:- Declare integer Global Variables queue[10] and f=-1, r=-1.

Step 3:- Stop.

UDF Enqueue ()

Step 1:- Start.

Step 2:- Declare integer variable “data” and n=sizeof(queue)/sizeof(queue[0]).

Step 3:- Read variable “data”.

Step 4:- Check **if** condition r==n-1, if it is true go to step 5 otherwise go to step 6.

Step 5:- Print Queue is Full.

Step 6:- Check **if** condition f==-1, if it is true go to step 7 otherwise go to step 8.

Step 7:- Increment f++.

Step 8:- Increment r++ and queue[r]=data

Step 9:- Stop.

UDF Dequeue ()

Step 1:- Start.

Step 2:- Check **IF** condition f==r, if it is true go to step 3 otherwise go to step 4.

Step 3:- Print Queue is Empty.

Step 4:- Declare “loc” variable and assign loc=queue[f] and Increment f++.

Step 5:- Print “loc” variable element has left the Queue.

Step 6:- Stop.

Main()

Step 1:- Start.

Step 2:- Declare ‘i’ variable for loop.

Step 3:- Call Enqueue operation 6 times.

Step 4:- Call Dequeue operation 2 times.

Step 5:- Print “queue” by loop.

Step 6:- Stop.

**Program** :-

#include<stdio.h>

int queue[10]; //Global Variables

int f=-1, r=-1;

void enqueue()

{

int data, n=sizeof(queue)/sizeof(queue[0]);

printf("Enter the Element in the Queue???\n");

scanf("%d",&data);

if(r==n-1)

{

printf("The Queue is Full\n");

}

else

{

if(f==-1)

f++;

r++;

queue[r]=data;

}

}

void dequeue()

{

int loc;

if(f==r)

printf("The Queue is Empty\n");

else

{

loc=queue[f];

f++;

printf("%d Element has left the Queue\n",loc);

}

}

int main()

{

int i;

enqueue();

enqueue();

enqueue();

enqueue();

enqueue();

enqueue();

dequeue();

dequeue();

printf("\n\nElements of Queue are....\n");

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

if(i>=f)

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

return 0;

}

**Output** :-

Enter the Element in the Queue???

1

Enter the Element in the Queue???

2

Enter the Element in the Queue???

3

Enter the Element in the Queue???

4

Enter the Element in the Queue???

5

Enter the Element in the Queue???

6

1 Element has left the Queue

2 Element has left the Queue

Elements of Queue are....

3

4

5

6

**Observation :-** After performing the experiment we observed that Values can be stored in a Queue and we were able to use UDF Enqueue and Dequeue operations. Insertion and Deletion of Values happened from two end only. The Insertion happens from last of the Queue and Deletion happens from the front of the Queue. It took 0.20 sec Compilation Time.