	PUNE INSTITUTE OF COMPUTER TECHNOLOGY PUNE - 411043	
	Department of Electronics & Telecommunication	
	ASSESMENT YEAR: 2021-2022	CLASS: SE-5
	SUBJECT: DATA STRUCTURES	
EXPT No:	LAB Ref: SE/2021-22/	Starting date: 22/11/2021
	Roll No:22108	Submission date:28/11/2021
Title:	**Stack and Queue Operations**	
Problem statement	Implement stack and queue using arrays. Perform push, pop, insert and delete operations on it.	
Prerequisites:	Basics of C programming	
	Decision making and loop controls	
	Choice based program	
	Functions, Stack and Queue	
Objectives:	Learn to create and display a stack and queue	
	Implement various operation on stack and queue to understand its effect on data.	
Theory:		

1) **Functions –**

- It is a self-contained block of statements that perform task of some kind.
- C program may have one or more functions
- C program must have at least one function i.e. main()
- There is no limit on number of functions
- Each function is called in a sequence specified by the function calls in main()
- After each function has done its job, control returns to next location from where it has been called.

2) **Array –**

- Arrays are used to store multiple values in a single variable. An array is a special variable, which can hold more than one value at a time.
- They are used to store similar type of elements as in the data type must be the same for all elements.
- Declared as – (datatype) (varname)[10], example – int arr[10]. Here, 10 is the limit of no. of elements predefined in the array.

3) **Stack –**

- It is an ordered group of homogeneous items of elements.
- Elements are added to and removed from the top of the stack (the most recently added items are at the top of the stack).
- Elements in between the stack cannot be removed or element cannot be added between the stack, for that all the elements above it needs to be removed and stored separately and then finally we will be able to 'pop' (remove/delete permanently) or 'push' (add element),
- The last element to be added is the first to be removed (LIFO: Last In, First Out).

4) **Queue** –

- It is an ordered group of homogeneous items of elements.
- Elements are added at one and removed from the other (at the end).
- The last element to be added is the first to be removed (FIFO: First In, First Out).

Code

```
Stack Operations –  #include<stdio.h>
int stk[100], choice, top, i, size, checker = 0;

void end()
{
   printf("\n==========This is the end of execution!==========");
return;
}

void display(int stk[100], int size, int top)
{
   if(top>=0)
   {
      printf("%d",top);
      printf("\n The elements in Stack:\n");
for(i=top; i>=0; i--)
      {
         printf("\n________\n");
printf("|   %d   |",stk[i]);
         printf("\n________");
      }
end();
   }
else
   {
      printf("\n The STACK is empty");
end();
   }
}

void push(int stk[100], int size)
{
   int ele;     if
(top>=size-1)
   {
      printf("\nPlease enter the size greater than 0");
return;     }
   for (i = 0; i < size; i++)
   {
top++;
      printf("Enter the %d element: ", top+1);
scanf("%d", &ele);         stk[top] = ele;
   }
```

```
    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0):
");     scanf("%d", &checker);
    if (checker == 1)
    {
        display(stk, size, top);

end();
return;
}     else
{
end();
return;
    }
}

void pop(int stk[100], int size, int top)
{
if(top<=-1)
    {
        printf("\n\t Stack is under flow");
    }
else
    {
        printf("\n\t The popped element is %d",stk[top]);
top--;
    }
    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0):
");     scanf("%d", &checker);
    if (checker == 1)
    {
        display(stk, size, top);

return;
}     else
{
        end();
return;
    }
}

void insert(int stk[100], int size, int top)
{     int ele;
if(top>=size-1)
    {
        printf("\n\tSTACK is over flow");

}
else
    {
        printf("Enter the element which needs to be pushed to the top of stack:
");          scanf("%d",&ele);
        top++;
stk[top]=ele;
    }
    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
    if (checker == 1)
    {
        display(stk, size, top);

return;
}     else
{
return;
    }
}

void delete(int stk[100], int size)
{
    int position, above[50], c = 0;
    printf("\nEnter the position of the element to be deleted: ");      scanf("%d",
&position);      printf("\n\nFirst, we will remove and store the elements above
the %d position element seperately, as we can't remove an element between a
stack without removing the elements above it!");     top = size-1;
    for (i = position-1; i < size; i++)
    {
        stk[top] = above[i];
        top--;
c++;
```

```
    }
    printf("\nThe elements above %d position element are: ");
    for (i = 0; i < c-1; i++)
    {
        printf("%d", above[i]);
    }
    printf("\nNow we will pop the element at %d position initially\n",
position);     pop(stk, size, top);
    printf("\nNow we will push all the elements that were above it.");

    for (i = 0; i < c; i++)
    {
top++;
        stk[top] = above[i];
    }

    printf("\nDo you wish the stack to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
    if (checker == 1)
    {
        display(stk, size, top);

return;
}     else
{
end();
return;
    }
}

int main()
{    top =
-1;
    printf("===============Stack
           Operations===============\nEnter the number of elements in the
stack (max = 100): ");     scanf("%d", &size);
    printf("When the elements will be entered and push to the stack.\n Enter
the elements one by one-\n");

    push(stk, size);

    printf("There are total 3 operations:\n 1)Pop\n 2)Insert\n 3)Delete\n
4)Exit\nEnter your choice: ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1: //Pop
    {
        pop(stk, size, top);
break;
    }
    case 2: //Insert
    {
        insert(stk, size, top);
break;
    }
    case 3: //Delete
    {
        delete(stk,
size);         break;
}
    case 4: //Exit
    {
end();
return 0;
    }
default:
    {
        printf("\nPlease enter a valid choice between numbers 1 to 4! Try
again!!");         break;
    }     }
return 0;
}
```

Queue Operations –

```
#include <stdio.h>
int q[100], choice, front, rear, i, size, checker = 0;

void end()
{
    printf("\n==========This is the end of execution!==========");
return;
}

void display(int q[100], int size, int rear)
{
 if (rear >= 0)
 {
  printf("\n The elements in Queue:\n");
  for (i = 0; i < rear; i++)
  {
   printf("%d\t", q[i]);
  }
end();
}  else
 {
  printf("\n The Queue is empty");
  end();
 }
}

void dequeue(int q[100], int size, int top)
{
 if (rear == -1 && front == -1 || front > rear)
 {
  printf("\n\t Queue is under flow.");

return;
}  else
 {
  printf("\n\t The popped element is %d", q[front]);
front++;
 }
 printf("\nDo you wish the Queue to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
 if (checker == 1)
 {
  display(q, size, top);

return;
}  else
{
end();
return;
 }
}

void enqueue(int q[100], int size, int rear)
{  int
ele;
```

```
   printf("Enter the element which needs to be pushed to the top of Queue: ");
scanf("%d", &ele);
   rear++;
q[rear] = ele;
   printf("\nDo you wish the Queue to be displayed? (y = 1/n = 0): ");
scanf("%d", &checker);
   if (checker == 1)
   {
      display(q, size, rear);
      return;
}    else
{
return;
   }
}
```

```
int main()
{     int ele;
front = -1;
rear = -1;
    printf("===============Queue
            Operations===============\nEnter the number of elements in the
Queue (max = 100): ");     scanf("%d", &size);
    printf("When the elements will be entered and push to the Queue.\n Enter the
elements one by one-\n");

    for (i = rear + 1; i < size; i++)
    {
rear++;
        scanf("%d", &ele);
q[rear] = ele;
    }

    printf("There are total 3 operations:\n 1)Insert\n 2)Delete\n 3)Exit\nEnter your
choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
    case 1: //Enqueue
    {
        enqueue(q, size, rear);
        break;
    }
    case 2: //Dequeue
    {
        dequeue(q, size, rear);
        break;
    }
    case 3: //Exit
    {
end();
return 0;
    }
default:
    {
        printf("\nPlease enter a valid choice between numbers 1 to 3! Try
again!!");         break;
    }
}
```

```
    return 0;
}
```

CONCLUSION:

In this experiment, we are able to implement stack and queue operations using array. We applied POP, Push, Insert and Delete in Stack and Insert and Delete in Queue.

REFERENCES:

Seymour Lipschutz, Data Structure with C, Schaum's Outlines, Tata McGrawHill

E Balgurusamy - Programming in ANSI C, Tata McGraw-Hill (Third Edition)

Yashavant Kanetkar- Let Us C, BPB Publication, 8th Edition.

Continuous Assessment for DS AY 2021-22			
RPP (5)	**SPO (5)**	**Total (10)**	**Signature:**
			Assessed By: Mr. V. B. Vaijapurkar
Start date	**Submission date**		**Date:**
22/11/2021	**28/11/2021**		**Roll. No.22108**
***Regularity, Punctuality, performance** ***Submission, Presentation, orals**			