**EXPERIMENT NO-4**

**Title:** Write a program to implement Stack and Queue data structures.

**Objectives:**

1. study of stack and queue data structures
2. implementation of stack and queue using object-oriented programming

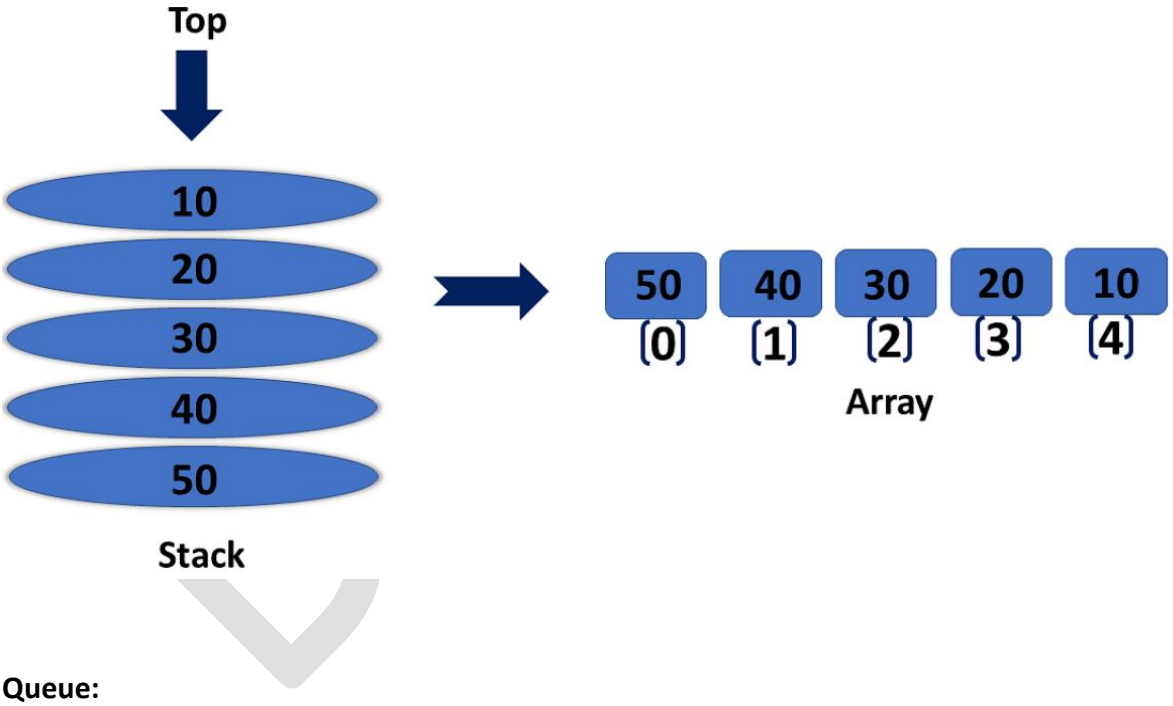
**Theory:**

**Stack:**

Stack is a data structure with ordered collection of data items in which elements are inserted from one end and deleted from the same end called as TOP of the stack.

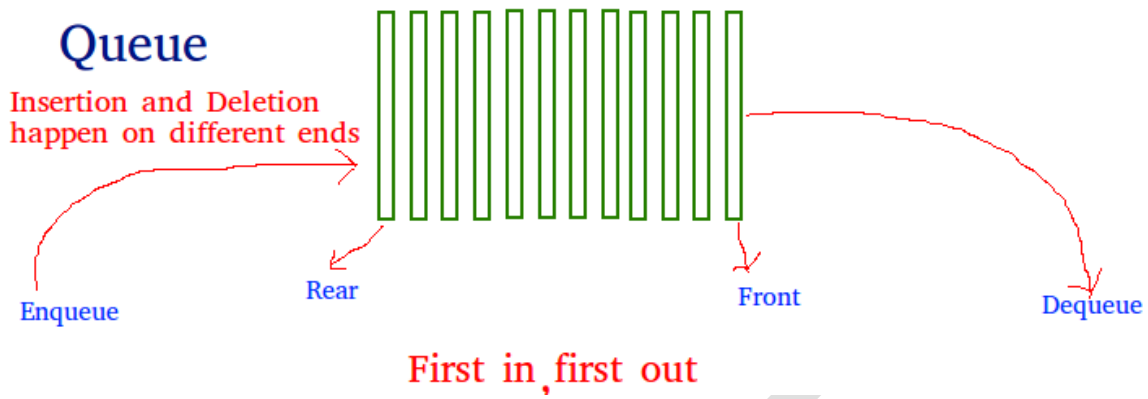
Insertion of elements is called as PUSH operation whereas removal of elements is called as POP operation. The element which is inserted at last is removed first. Hence, it is called as 'LIFO' (Last In First Out).

If elements are inserted more than the size of the stack, then it is called as "stack overflow". If elements are deleted from empty stack, then it is called as "stack underflow".




**Queue:**

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.




**Procedure:**

1. Write a Stack class with following members.

Data Members of Stacks

*stack_array, top and size

Provide following public operations on Stack

Overload constructor: stack(), stack(size) - to initialize stack size and allocate memory

push(data) – store data in the stack and return status

pop - return stacked data from the top of the stack.

Isempty() – return 1 if stack is empty, 0 otherwise.

Isfull() – return 1 of stack is full, 0 otherwise.

Destructor- to deallocate memory

2. Write Circular Queue Class with following members.

Data members of queue:

*queue_array, front, rear, space_used, size

Provide following public operations on the queue:

Overloaded constructors:

Queue() – allocates fixed size of memory for queue

Queue() – allocates memory as specified by user.

Destructor:

~Queue() – Deallocates dynamically allocated memory

Other public members:

enqueue – store the data in the queue at rear position and return the status

dequeue – returns the data from the front position.

isFull - returns 1 if the queue is full, 0 otherwise

isEmpty – returns 1 if the queue is empty, 0 otherwise

3. Create the object of Stack and Queue Classes and test its functionality.

**Keywords: stack, queue, overloaded constructors, destructor.**