Lab No.10 Stacks and Queues

10.1 Objectives of the lab:

Introducing the concepts of some linear data structures such as

38 Stack
39 Linear Queue
40 Circular Queue

10.2 Pre-Lab

10.2.1 Stack

1 Stores arbitrary objects
2 Insertions and deletions follow the Last-in-First-out scheme
3 Also known as LIFO or FILO structure
4 Main Stack Operations
- Push (Object o): Inserts element o
- Pop (Object o): Removes element o

5 Auxiliary stack operations
- Top (): Returns the last inserted element without removing it.
- Size(): Returns the number of stored elements
- IsEmpty(): A Boolean value indicating whether no elements are stored

10.2.2 Stack Algorithms

We have two algorithms for stack:

1 Insertion called as **PUSH** operation: add an item to the top of the Stack
2 Deletion called as **POP** operation: delete an item from the stack

10.2.3 Algorithm for PUSH

1. [Stack is full already?]
   If Top= MAXSTK, then print: Overflow, and Return.

2. Set TOP= TOP +1.
3. SET stack (TOP)=item[Inserts ITEM in new position.]
4. End

10.2.4 Algorithm for POP operation

1. Check for underflow i.e.
   If (Top= = 0)
   Print underflow and return
2. assign item to variable i.e.
   var =stack(TOP)

3. Decrement top by one top=top-1
4. End

10.3 Queue

1 Stores arbitrary objects
2 Insertions and deletions follow the First In First Out scheme
3 Insertions are at the rear of the queue and deletions at the front
4 Main Queue Operations
  - Enqueue(Object o): Inserts element o, at the rear of the queue
  - Dequeue(): Removes & returns the element from the front of the queue
5 Auxiliary Queue Operations
  - Front(): Returns the element at the front without removing it
  - Size(): Returns the number of elements stored
  - IsEmpty(): Returns a boolean value indicating whether the queue is empty
6 Exceptions
  - Attempting the dequeue and Front operations on an empty queue

10.3.1 Queue Algorithms

We have two algorithms for queue:

3 Insertion called as **Enqueue** operation: add an item to the rear of the queue
4 Deletion called as **Dequeue** operation: remove an item from front of queue

10.3.2 Algorithm for Enqueue

1. [Queue is full already?]
   If FRONT= 1 and REAR=N, or if FRONT=REAR+1, then print: Overflow, and Return.
2. Find new value of REAR i.e.
   If FRONT= NULL, then set FRONT +REAR+1
   Else if REAR= N, then
   Set REAR=1
   ELSE
   Set REAR=REAR+1.
3. SET QUEUE [REAR] =item [Inserts ITEM in new position.]
4. End

10.3.3 Algorithm for Dequeue

1. Check for underflow i.e.
   If (FRONT = = 0)
   Print underflow and return
2. assign item to variable i.e.
   var =QUEUE[FRONT]
3. [Find new value for FRONT]
   If FRONT =REAR, then

```
            SET FRONT = REAR= NULL
    ELSE if FRONT =N, then
                SET FRONT =1
    ELSE
                SET FRONT= FRONT +1
```

4. End

10.4 In-Lab

10.4.1 Activity

Understand code of array based Stack. Add auxiliary Stack operations.

10.4.2 Activity

Understand code of array based Queue. Add auxiliary Queue operations.

10.5 Home-Lab

10.4.3 Activity

Create a structure for **book** that contains information about title, price, edition, and no of pages of the book. Use your program of activity 1 to push and pop books on a stack.

10.4.4 Activity

Let us model the flow of customers in a queue. Create a structure called **Person** that contains information about first name, last name, age, sex, and address of a person. Use your program of activity 2 to enqueue and dequeue Persons from a queue.

10.4.5 Activity

Implement a circular-queue whose front and rear are connected together. Justify what is the problem in linear queue due to which we need a circular queue.

10.4.6 Activity

Implement a list using array with following operations:

1 Traversal
2 Insertion at beginning
3 Insertion at end
4 Insertion at nth location
5 Deletion from beginning
6 Deletion from end

7 Deletion from nth location

8 Searching an element

10.6 References

27 Class notes

28 Data Structures, Schaum's outline series

29 **Data structures and algorithms in C++ by *Micheal T. Goodrich and Roberto Tamassia***