Basic Stack and Queue

❖ 3.1Linear Data Structure and Non Linear Data Structure

1) Linear Data Structure:

- ✓ Logical relation among the data item is linear.
- ✓ Data is stored sequentially in memory location.
- ✓ Examples of the linear data structure are:

(A) Array (B) Stack (C) Queue (D) Linked List

2) Non Linear data structure:

- ✓ Logical relation among the data item is not linear.
- ✓ Data is not stored sequentially.
- ✓ Examples of the non linier data structure are:(A)Tree

(B) Graph

❖ 3.2 Stack

- ✓ **A Stack is a linear OR non-primitive list in which insertion and deletion operations are performed at only one end of the list.**
- ✓ **A Stack is Last in First out (LIFO**) Structure.
- ✓ A stack is a linear data structure in which elements are added and remove from one end, called top of stack.
- ✓ For example of stack is, consider a stack of plates on the counter in cafeteria, during the time of dinner, customer take plates from top of the stack and waiter puts the washed plates on the top of the stack. So new plates are put on the top and old one yet bottom.

Write an algorithm for PUSH operation of Stack.

- ✓ In push operation, we can add elements to the top of the stack, so before push operation, user must check the stack, it should not be a full.
- ✓ If stack is already full and when we try to add the elements then error occurs. It is called “Stack Over Flow” error.

Algorithm: PUSH(S, TOP, X, N)

S :→ S is a vector which contains N elements.

N :→ Total no of elements in a stack.

TOP :→ TOP is a pointer which points to the top element of the stack.

```
Step-1: [Initialization]
        TOP← -1
Step-2: [Check for stack overflow]
            If (TOP >= N-1) then
                Write ("Stack Overflow")
                Exit
Step-3: [Increment TOP pointer by value 1]
        TOP ← TOP + 1
Step-4: [Insert Element]
        S [TOP] ← VAL
Step-5: [Finished]
        Return
```

Write an algorithm for POP operation of Stack.

- ✓ In pop operation, we can delete or remove an element from top of stack.
- ✓ User must check the stack, stack should not be a empty.
- ✓ If a stack is empty, and we can try to delete an element, then a stack underflow erroroccurs.

Algorithm: POP(S,TOP,N)

S :→ S is a vector which contains N elements.

N :→ Total no of elements in a stack.

TOP :→ TOP is a pointer which points to the top element of the stack.

```
Step-1: [Check for stack underflow or check whether the stack is empty]
            If (TOP = -1) then
                Write ("Stack Underflow")
                Exit.
Step-2: [Remove top element and decrement pointer]
                X ← S [TOP]
                TOP ← TOP - 1
        Step-3: [Return top element of the stack]
              Write (X)
          Step-4: [Finished]
             Return
```

Infix, Prefix and Postfix Forms of Expressions

- ✓ There are basically three types of polish notation:
- ✓ Infix (B) Prefix (C) Postfix
- ✓ When the operator(+) exists between two operands(A & B) then it is known as Infix notation.(e.g. A+B)
- ✓ When the operator(+) are written before their operands(A & B) then it is known as Prefix notation(Polish notation). (e.g. +AB)
- ✓ When the operator(+) are written after their operands(A & B) then it is known as Postfix notation(Reverse polish notation). (e.g. AB+).
- ✓ Stack is widely used to convert infix notation into prefix or postfix notation.

<u>Rules for converting infix notation to prefix/ postfix</u>

- ✓ The operations with high precedence are converted first and then after a portion of the expression have been converted to postfix.
- ✓ It is to be treated as single operand.
- ✓ Take only two operands at a time to convert in a postfix from like A+B→ AB+
- ✓ Always, convert the parenthesis first
- ✓ Convert postfix exponentiation second if there are more than one exponentiation insequence then take order from right to left.
- ✓ Convert in to postfix as multiplication and division operator, left to right.
- ✓ Convert in postfix as addition and subtraction left to right.

Convert infix into Prefix expression.

1. **(A + B) * C** = (+ A B) * C = * (+ A B) C = * + A B C	**2. (A + B) * (C + D)** = (+ A B) * (+ C D) =*(+ A B) (+ C D) =* + AB + CD	**3. (A-B)+C*A+B** =-AB+C*A+B =-AB+*CA+B =+-AB*CA+B =++-AB*CAB
4. (B*C/D)/(D/C+E) =(*BC/D)/(D/C+E) =(/*BCD)/(D/C+E) =(/*BCD)/(/DC+E) =(/*BCD)/(+/DCE) =//*BCD+/DCE	**5. A*B/(C+D-E)** =A*B/(+CD-E) =A*B/(-+CDE) =*AB/-+CDE =/*AB-+CDE	

Convert infix into Postfix expression.

A * B / C +D = [AB*]/C+D = [AB*C/] +D =AB*C/D+	2. C*B*A/D = [CB*]*A/D = [CB*A*]/D =CB*A*D/	3. (A-B)+C*A+B = [AB-] +C*A+B = [AB-] + [CA*] +B = [AB-CA*+] +B =AB-CA*+B+
4. A*B/(C+D-E) =A*B/([CD+]-E) =A*B/[CD+E-] =[AB*]/[CD+E-] =AB*CD+E-/	5. (B*C/D)/(D/C+E) = ([BC*]/D)/(DC/+E) = [BC*D/]/[DC/E+] =BC*D/DC/E+/	6. A*B+C*D+E*F =[AB*]+C*D+E*F =[AB*]+[CD*]+E*F =[AB*]+[CD*]+[EF*] =[AB*CD*+]+[EF*] =AB*CD*+EF*+

Write an application of Stack.

(1) Recursion:

- ✓ Recursion means function call itself.
- ✓ Stack is widely used in recursion because of its Last in First out Property.
- ✓ There are two types of recursion:
    1. Primitive Recursive Function
    2. Non Primitive Recursive Function

(2)Evaluate polish notation:

- ✓ The process of writing the operators of an expression either before their operands or afteroperands are called the polish notation.
- ✓ There are three types of polish notation: Infix, Prefix and Postfix

(2) Stack Machine:

- ✓ It is used for executing reverse polish notation.
- ✓ It is also used for stacking local variables.
- ✓ Stack machines provides faster execution of polish notation.
- ✓ Computer performs stack operation at the hardware or machine level.
- ✓ One problem with stack machine is that it has very limited number of registers.

Recursive Functions (factorial, Fibonacci series)

- ✓ Recursion means a function call itself.

Advantage of recursion:

- Simple to implement
- It is very useful when the solution of a problem is in repetitive form.
- It is used with data structure like:stack,queue,linked list etc.

Disadvantage of recursion:

- Care should be taken else chance of infinite loop.
- We have to solve sub problems recursively.

Example: The **factorial function** can be recursively defined as,

- ✓ Here, FACTORIAL(N) is defined in terms of FACTORIAL(N-1) which is again defined in terms of FACTORIAL(N-2) and this process continue until FACTORIAL(0) is reached.
- ✓ The value of FACTORIAL (0) is defined as 1. Example: To

Generate the **Fibonacci number** using recursion:

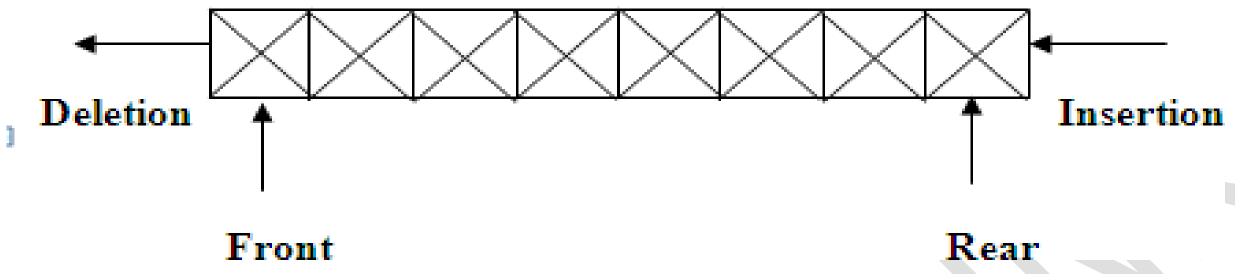
- ✓ A fibonacci series is,

0 1 1 2 3 5 8........................ N

❖ 3.4 Queue

- ✓ **A queue is a linear list in which insertion is performed at one end called rear end and deletion is performed at another end of the list called front end.**

- ✓ **The information in such a list is proceeds in the same order as it was received.**
- ✓ Since insertion is performed at one end and deletion is performed at another end the element which is inserted first is first to delete. So it is also known as *First in First out (FIFO)* or *First Come First Serve (FCFS)* data structure.



Write an algorithm for insert operation of Queue.

- ✓ This function insert an element into the queue.
- ✓ In Insert operation, we can add elements to the rear end of the queue, so before insert operation, user must check the Queue, it should not be a full.
- ✓ If Queue is already full and when we try to add the elements then error occurs. It is called"Queue Over Flow" error.

ALGORITHM: QINSERT (Q,X,front,rear,N)

- ✓ The Queue is represented by vector Q which contains N elements.
- ✓ Front is a pointer which points to the front end.
- ✓ Rear is a pointer which points to the rear end.
- ✓ X is the element to be inserted.

```
Step-1: [Check Overflow error?]
        If Rear >= N-1 then
            Write ("Queue Overflow")
            Exit
Step-2: [Increment rear pointer]
        Rear←Rear+1
Step-3: [Insert element]
        Q [Rear]←X
Step-4: [Is front pointer properly set?]
        If Front = -1 then
        Front←0
    Step-5: [Finished]
            Exit.
```

Write an algorithm for delete operation of Queue.

- ✓ Delete operation delete an element fro, a queue at front end and then increment frontpointer variable by one.
- ✓ In delete operation, we can delete or remove an element from front end of Queue.
- ✓ User must check the Queue, Queue should not be a empty.
- ✓ If a Queue is empty, and we can try to delete an element, then a Queue underflow erroroccurs.

ALGORITHM: QDELETE (Q, Front, Rear)

- ✓ The Queue is represented by vector Q which contains N elements.
- ✓ front is a pointer which points to the front end.
- ✓ rear is a pointer which points to the rear end.

```
Step-1: [Check for queue Underflow]
            If Front = -1 then
                Write ("Queue Underflow")
            Exit
    Step-2: [Delete element]
            X←Q [front]
    Step-3: [Write the deleted Element]
          Write ("X")
    Step-4: [Check for queue empty]
            If front = rear then
                front ← -1
                rear ← -1
            else
```

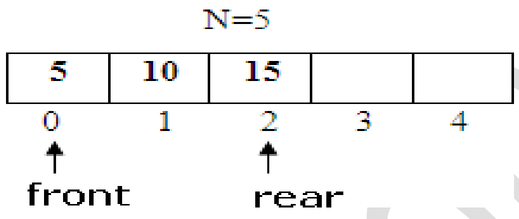
```
                front ←front +1
        Step-5: [Finished]
                Exit.
```

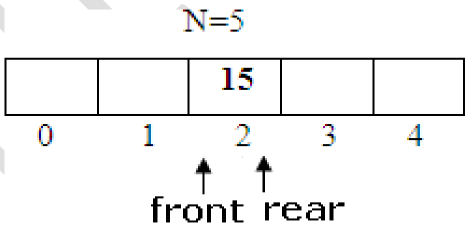
Write down applications of queue.

- ✓ Queue is widely used in Simulation.
- ✓ In OS, Process scheduling or disk scheduling algorithm use queue concept.
- ✓ Processes in a system is work according to queue.
- ✓ A Computer Processor also maintain buffer in the form of queue for incoming resourcerequest.

❖ **Limitation of Simple Queue.**

- ✓ Disadvantage of simple queue is that even if we have a free memory space in a queue we cannot use that free memory space to insert element.
- ✓ For example consider following operations:

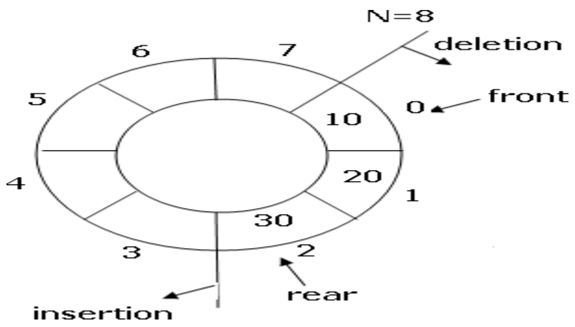




- ✓ As shown in figure we insert three elements 5, 10, 15 in simple queue. After that we delete 5and 10 as shown in figure. Now even we have a free memory space we cannot use that memory space. So simple queue results in wastage of memory space. This problem can be solved by using circular queue.

❖ **Write a short note on Circular Queue.**

- ✓ A circular queue is a queue in which elements are arranged such that the first element in the queue follows the last element in the queue.
- ✓ The arrangement of circular queue is shown in figure below:



- ✓ Here in the circular queue the first element q[0] follows the last element q[n-1].
- ✓ Disadvantage of simple queue is that even if we have a free memory space in a queue we cannot use that free memory space to insert element. Circular Queue overcomes limitation of Simple queue.

Difference between LIFO and FIFO.

LIFO	FIFO
In LIFO the insertion and deletion operation are performed at only one end.	In FIFO the insertion and deletion operation are performed at two different ends.
In LIFO the element which is inserted last is first to delete.	In FIFO the element which is inserted first is first to delete.
LIFO require only one pointer called TOP	FIFO requires two pointers called front and rear.
Example: piles of trays in cafeteria	Example: students at registration counter
In LIFO there is no wastage of memory space.	In FIFO even if we have free memory space sometimes we cannot use that space to store elements.
Stack follow LIFO structure.	Queue follow FIFO structure.

Difference between Simple Queue and Circular Queue.

Simple Queue	Circular Queue
The last element of the queue does not follow the first element	The last element in the queue followthe first element.
Problem of Overflow in a simple queue is frequently occurs.	Problem of Overflow in a circular queue infrequently occurs.
There is wastage of Memory space.	There is no wastage of Memory space.