STACK QUEUE

- Defining Stack
- Implementing Stack
- Defining Queue
- Implementing Queue
- Application of Queue and Stack

Data Structure: The way of storage, management data so that efficient and optimum utilization of data can be achieved in order to information generation.

Stack

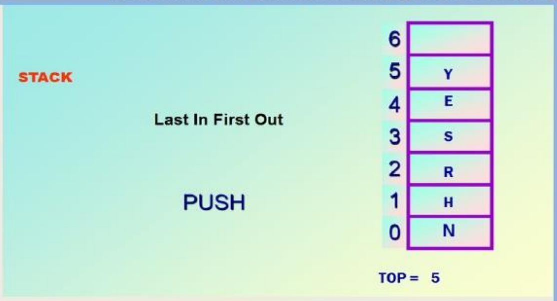
(i) **It is implemented as LIFO (Last in first out)**

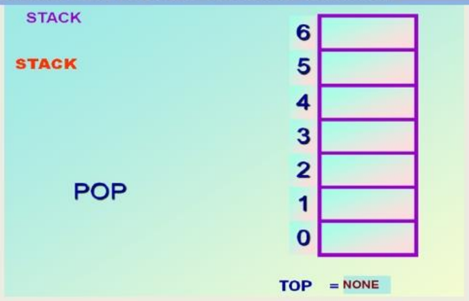
(ii) **It is linear data structure.**

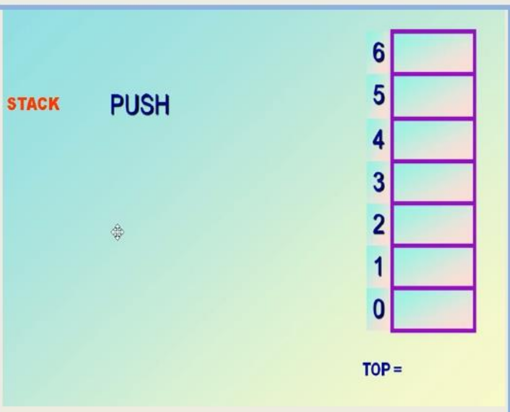
(iii) **Insertion and deletion of items takes place at one end called top of the stack**

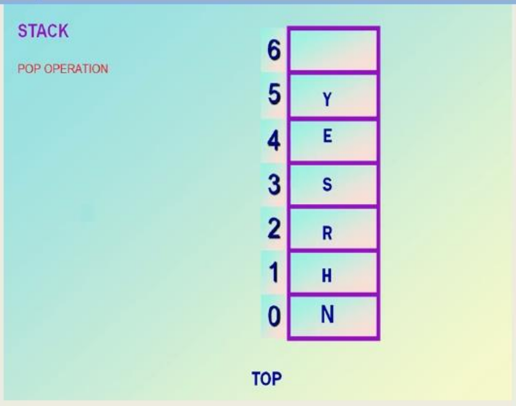
(iv) **Insertion is known as PUSH and deletion is termed as POP.**

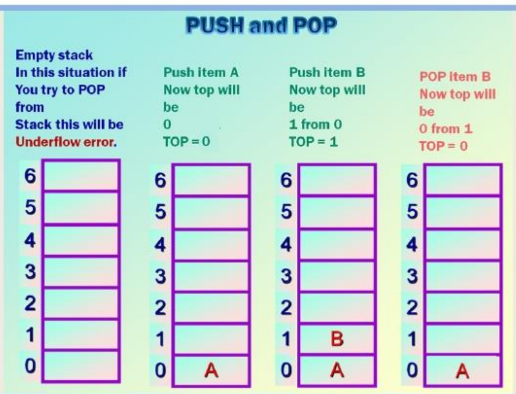
(v) **Overflow Underflow are error conditions, when stack size is limited and PUSH operations are performed it is overflow and when list is empty and deletion is performed it is known as Underflow condition.**

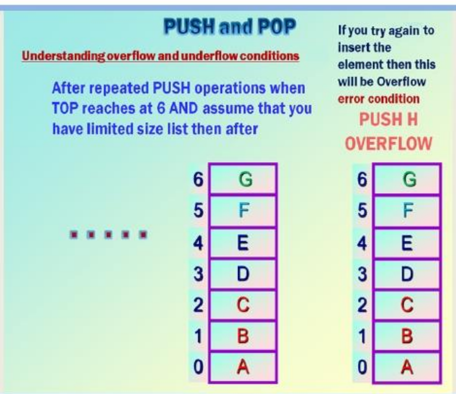




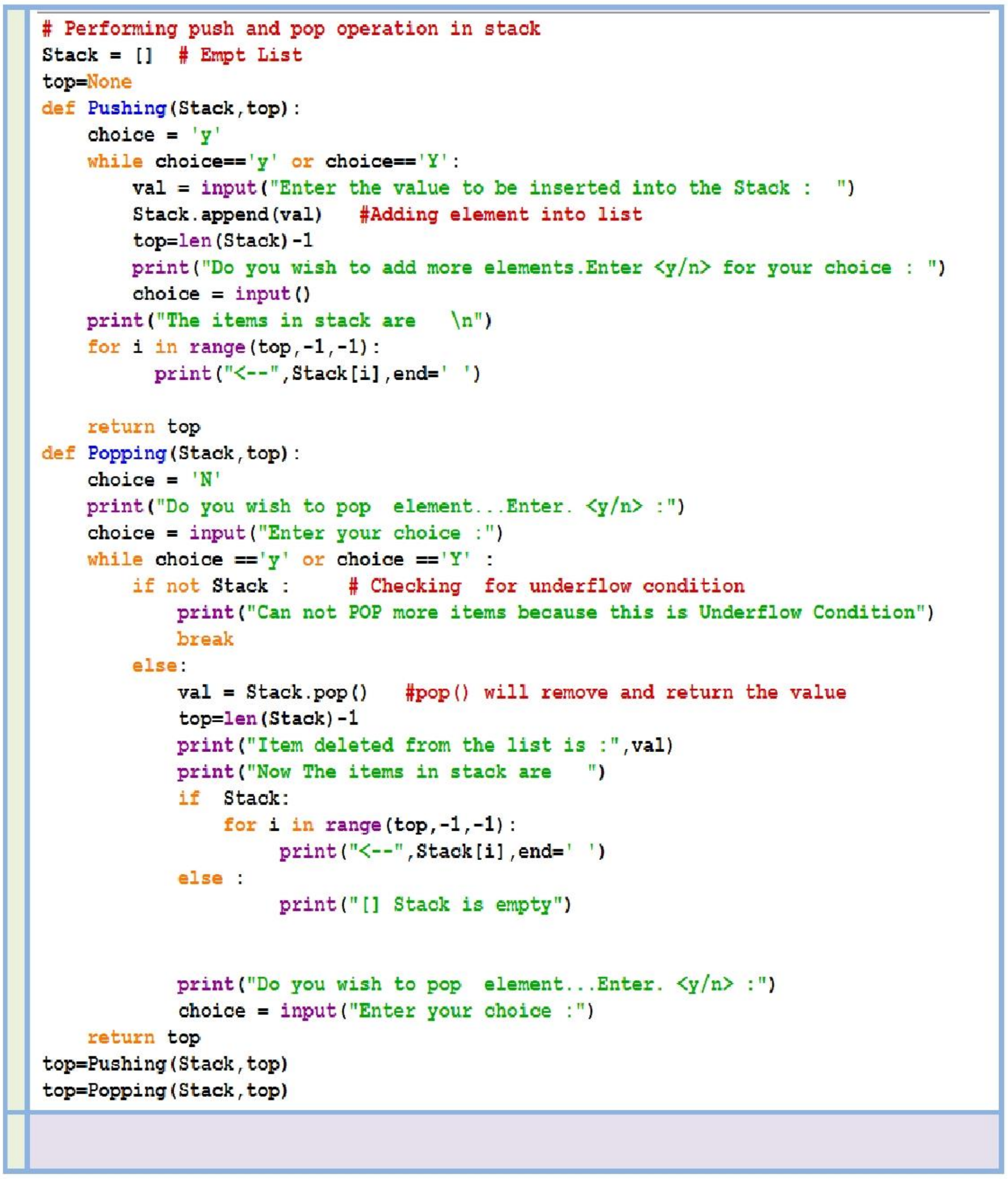




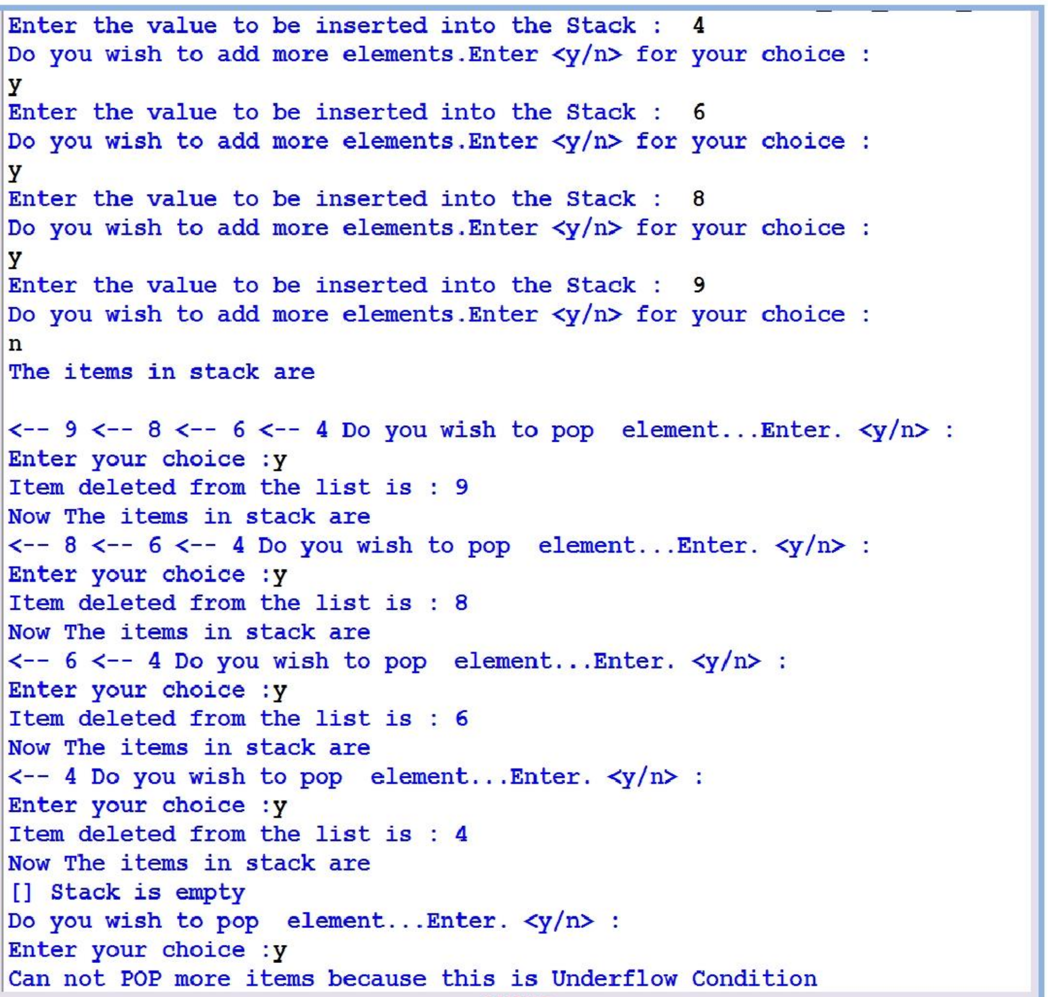



PUSH and POP
Understanding overflow and underflow conditions
After repeated PUSH operations when TOP reaches at 6 AND assume that you have limited size list then after
If you try again to insert the element then this will be Overflow error condition
PUSH H
OVERFLOW

For implementing the stack push and pop operations by python we shall use stack append () and pop () functions.

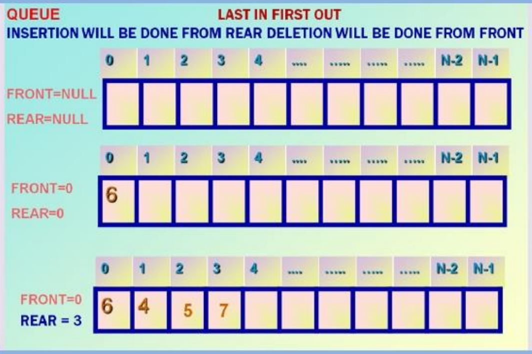


```
# Performing push and pop operation in stack
Stack = []  # Empt List
top=None
def Pushing(Stack,top):
    choice = 'y'
    while choice=='y' or choice=='Y':
        val = input("Enter the value to be inserted into the Stack :  ")
        Stack.append(val)   #Adding element into list
        top=len(Stack)-1
        print("Do you wish to add more elements.Enter <y/n> for your choice : ")
        choice = input()
    print("The items in stack are   \n")
    for i in range(top,-1,-1):
          print("<--",Stack[i],end=' ')

    return top
def Popping(Stack,top):
    choice = 'N'
    print("Do you wish to pop  element...Enter. <y/n> :")
    choice = input("Enter your choice :")
    while choice =='y' or choice =='Y' :
        if not Stack :     # Checking  for underflow condition
            print("Can not POP more items because this is Underflow Condition")
            break
        else:
            val = Stack.pop()   #pop() will remove and return the value
            top=len(Stack)-1
            print("Item deleted from the list is :",val)
            print("Now The items in stack are    ")
            if  Stack:
                for i in range(top,-1,-1):
                     print("<--",Stack[i],end=' ')
            else :
                     print("[] Stack is empty")


            print("Do you wish to pop  element...Enter. <y/n> :")
            choice = input("Enter your choice :")
    return top
top=Pushing(Stack,top)
top=Popping(Stack,top)
```



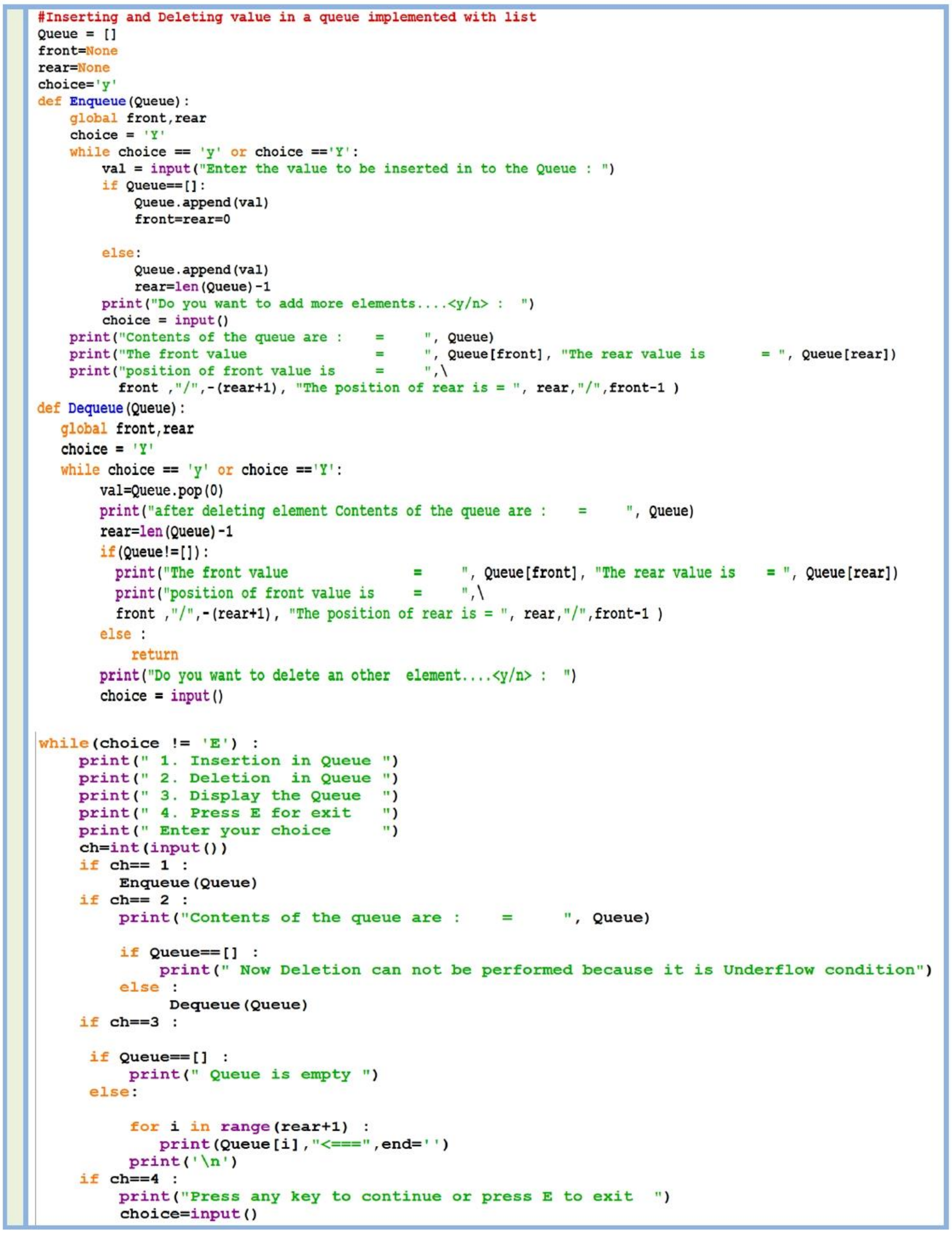
```
Enter the value to be inserted into the Stack :  4
Do you wish to add more elements.Enter <y/n> for your choice :
y
Enter the value to be inserted into the Stack :  6
Do you wish to add more elements.Enter <y/n> for your choice :
y
Enter the value to be inserted into the Stack :  8
Do you wish to add more elements.Enter <y/n> for your choice :
y
Enter the value to be inserted into the Stack :  9
Do you wish to add more elements.Enter <y/n> for your choice :
n
The items in stack are

<-- 9 <-- 8 <-- 6 <-- 4 Do you wish to pop  element...Enter. <y/n> :
Enter your choice :y
Item deleted from the list is : 9
Now The items in stack are
<-- 8 <-- 6 <-- 4 Do you wish to pop  element...Enter. <y/n> :
Enter your choice :y
Item deleted from the list is : 8
Now The items in stack are
<-- 6 <-- 4 Do you wish to pop  element...Enter. <y/n> :
Enter your choice :y
Item deleted from the list is : 6
Now The items in stack are
<-- 4 Do you wish to pop  element...Enter. <y/n> :
Enter your choice :y
Item deleted from the list is : 4
Now The items in stack are
[] Stack is empty
Do you wish to pop  element...Enter. <y/n> :
Enter your choice :y
Can not POP more items because this is Underflow Condition
```

QUEUE

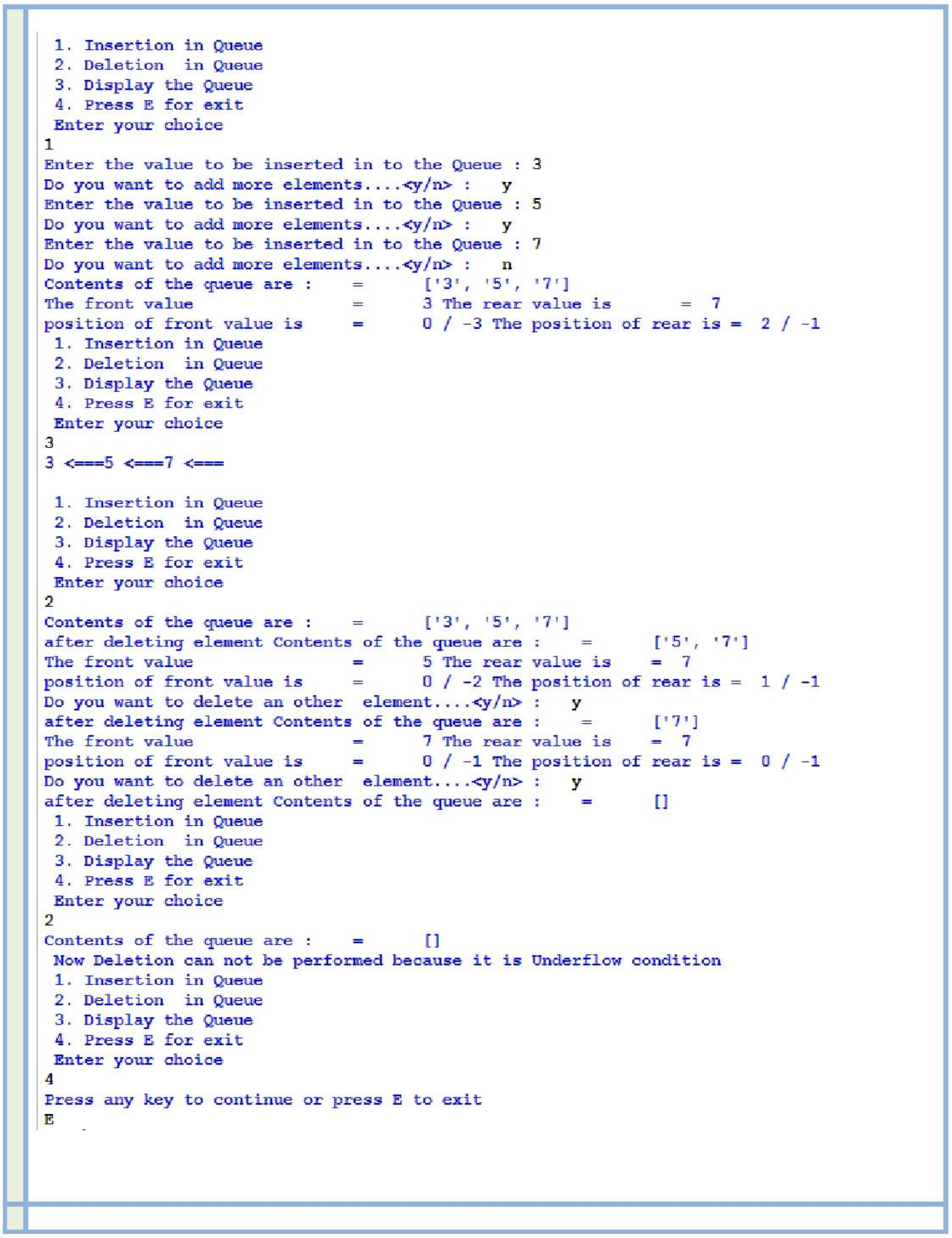

QUEUE
LAST IN FIRST OUT
INSERTION WILL BE DONE FROM REAR DELETION WILL BE DONE FROM FRONT
0 1 2 3 4 .... N-2 N-1
FRONT=NULL
REAR=NULL
FRONT=0
REAR=0
6
FRONT=0
REAR = 3
6 4 5 7

QUEUE

- It is implemented as FIFO (First in First out)
- Real Life example : Token system at Post office / Hospital for customers/patients
- Railway ticket window queue.
- Insertion in queue is known as Enqueue and Deletion known as Dequeue



```
#Inserting and Deleting value in a queue implemented with list
Queue = []
front=None
rear=None
choice='y'
def Enqueue(Queue):
    global front,rear
    choice = 'Y'
    while choice == 'y' or choice =='Y':
        val = input("Enter the value to be inserted in to the Queue : ")
        if Queue==[]:
            Queue.append(val)
            front=rear=0

        else:
            Queue.append(val)
            rear=len(Queue)-1
        print("Do you want to add more elements....<y/n> :  ")
        choice = input()
    print("Contents of the queue are :    =     ", Queue)
    print("The front value               =     ", Queue[front], "The rear value is       = ", Queue[rear])
    print("position of front value is     =     ",\
          front ,"/",-(rear+1), "The position of rear is = ", rear,"/",front-1 )
def Dequeue(Queue):
   global front,rear
   choice = 'Y'
   while choice == 'y' or choice =='Y':
        val=Queue.pop(0)
        print("after deleting element Contents of the queue are :    =     ", Queue)
        rear=len(Queue)-1
        if(Queue!=[]):
          print("The front value               =     ", Queue[front], "The rear value is    = ", Queue[rear])
          print("position of front value is     =     ",\
          front ,"/",-(rear+1), "The position of rear is = ", rear,"/",front-1 )
        else :
            return
        print("Do you want to delete an other  element....<y/n> :  ")
        choice = input()

while(choice != 'E') :
    print(" 1. Insertion in Queue ")
    print(" 2. Deletion  in Queue ")
    print(" 3. Display the Queue  ")
    print(" 4. Press E for exit   ")
    print(" Enter your choice     ")
    ch=int(input())
    if ch== 1 :
        Enqueue(Queue)
    if ch== 2 :
        print("Contents of the queue are :    =     ", Queue)

        if Queue==[] :
            print(" Now Deletion can not be performed because it is Underflow condition")
        else :
             Dequeue(Queue)
    if ch==3 :

     if Queue==[] :
         print(" Queue is empty ")
     else:

         for i in range(rear+1) :
            print(Queue[i],"<===",end='')
         print('\n')
    if ch==4 :
        print("Press any key to continue or press E to exit  ")
        choice=input()
```



```
 1. Insertion in Queue
 2. Deletion  in Queue
 3. Display the Queue
 4. Press E for exit
 Enter your choice
1
Enter the value to be inserted in to the Queue : 3
Do you want to add more elements....<y/n> :   y
Enter the value to be inserted in to the Queue : 5
Do you want to add more elements....<y/n> :   y
Enter the value to be inserted in to the Queue : 7
Do you want to add more elements....<y/n> :   n
Contents of the queue are :     =      ['3', '5', '7']
The front value                 =      3 The rear value is       =  7
position of front value is      =      0 / -3 The position of rear is =  2 / -1
 1. Insertion in Queue
 2. Deletion  in Queue
 3. Display the Queue
 4. Press E for exit
 Enter your choice
3
3 <===5 <===7 <===

 1. Insertion in Queue
 2. Deletion  in Queue
 3. Display the Queue
 4. Press E for exit
 Enter your choice
2
Contents of the queue are :     =      ['3', '5', '7']
after deleting element Contents of the queue are :    =      ['5', '7']
The front value                 =      5 The rear value is    =  7
position of front value is      =      0 / -2 The position of rear is =  1 / -1
Do you want to delete an other  element....<y/n> :   y
after deleting element Contents of the queue are :    =      ['7']
The front value                 =      7 The rear value is    =  7
position of front value is      =      0 / -1 The position of rear is =  0 / -1
Do you want to delete an other  element....<y/n> :   y
after deleting element Contents of the queue are :    =      []
 1. Insertion in Queue
 2. Deletion  in Queue
 3. Display the Queue
 4. Press E for exit
 Enter your choice
2
Contents of the queue are :     =      []
 Now Deletion can not be performed because it is Underflow condition
 1. Insertion in Queue
 2. Deletion  in Queue
 3. Display the Queue
 4. Press E for exit
 Enter your choice
4
Press any key to continue or press E to exit
E
```

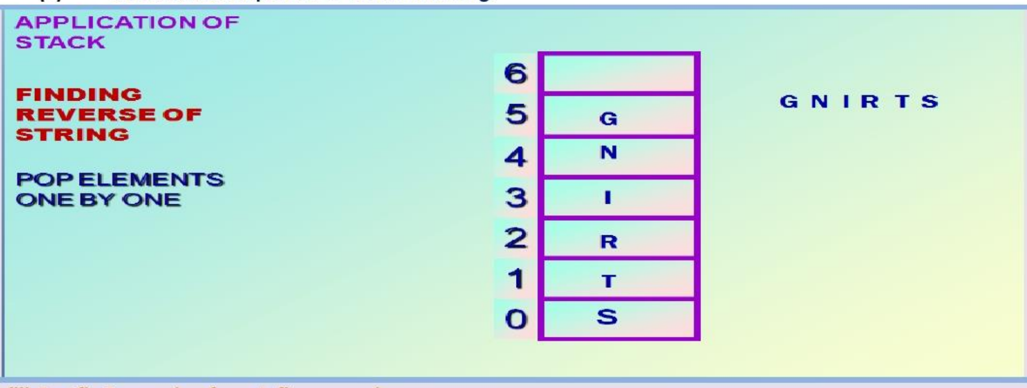
APPLICATION OF QUEUE

- **Used in Operating System for implementation of process scheduling like Round Robin.**
- Accessing shared resources like printers, disk.

APPLICATION OF STACK

(i) **Finding Reverse of string**

(ii) **It is used to keep track of function calling.**

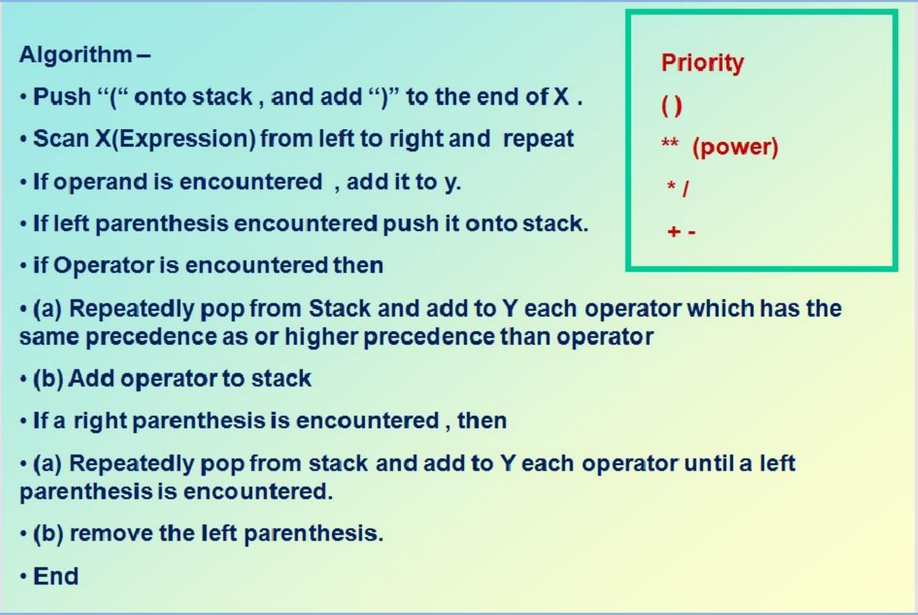


APPLICATION OF STACK

FINDING REVERSE OF STRING

POP ELEMENTS ONE BY ONE

(ii) Postfix Expression from Infix expression.



Algorithm –

- **Push “(“ onto stack , and add “)” to the end of X .**
- **Scan X(Expression) from left to right and repeat**
- **If operand is encountered , add it to y.**
- **If left parenthesis encountered push it onto stack.**
- **if Operator is encountered then**
- **(a) Repeatedly pop from Stack and add to Y each operator which has the same precedence as or higher precedence than operator**
- **(b) Add operator to stack**
- **If a right parenthesis is encountered , then**
- **(a) Repeatedly pop from stack and add to Y each operator until a left parenthesis is encountered.**
- **(b) remove the left parenthesis.**
- **End**

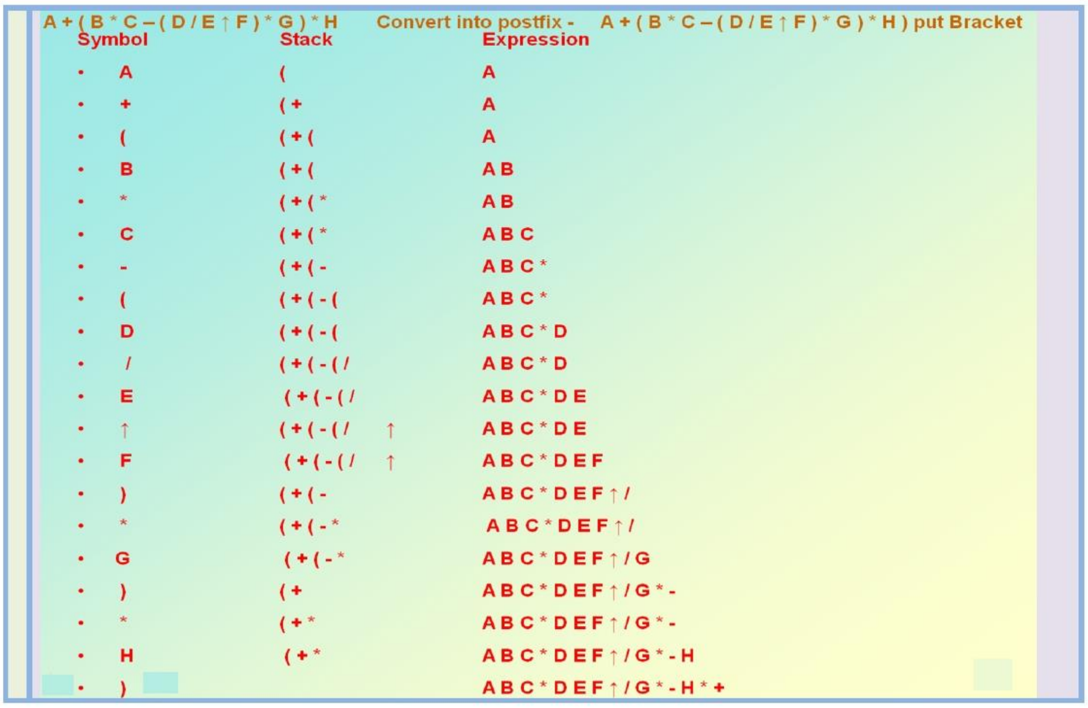
Priority

( )

**** (power)**

*** /**

+ -



A + (B * C – (D / E ↑ F) * G) * H   Convert into postfix -   A + (B * C – (D / E ↑ F) * G) * H) put Bracket

Symbol	Stack	Expression
• A	(	A
• +	(+	A
• (	(+ (	A
• B	(+ (	A B
• *	(+ (*	A B
• C	(+ (*	A B C
• -	(+ (-	A B C *
• (	(+ (- (	A B C *
• D	(+ (- (	A B C * D
• /	(+ (- (/	A B C * D
• E	(+ (- (/	A B C * D E
• ↑	(+ (- (/ ↑	A B C * D E
• F	(+ (- (/ ↑	A B C * D E F
•)	(+ (-	A B C * D E F ↑ /
• *	(+ (- *	A B C * D E F ↑ /
• G	(+ (- *	A B C * D E F ↑ / G
•)	(+	A B C * D E F ↑ / G * -
• *	(+ *	A B C * D E F ↑ / G * -
• H	(+ *	A B C * D E F ↑ / G * - H
•)		A B C * D E F ↑ / G * - H * +