COMPUTER SCIENCE AND DA

Data Structures through Python

Queues and Hash Tables

Lecture No. 1

By- Kashif Sir

RECAP

Expression Conversion

- Prefix to Postfix
- Postfix to Prefix
- Prefix to Infix
- Postfix to Infix

Expression Evaluation

TOPICS TO BE COVERED

- Problem solving
- Queue
- Operations
- Type of Queue

PROBLEM SOLVING

2) The following postfix expression with single digit operands is evaluated using a stack:

8 2 3 ^ / 2 3 * + 5 1 *

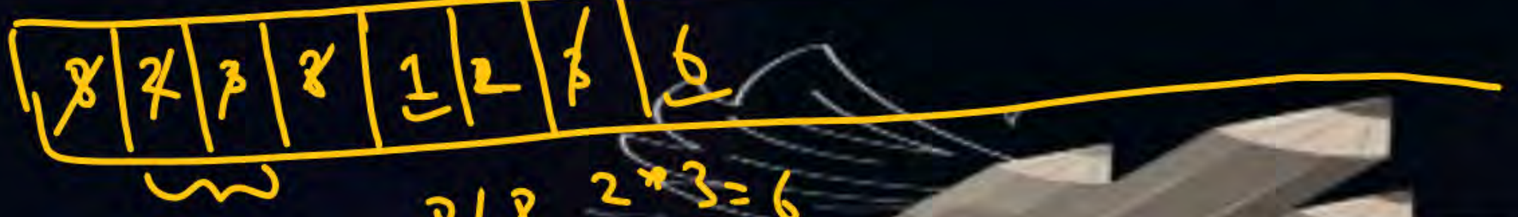
Note that ^ is the exponent operator. The top two elements of the stack after the first * is evaluated are:

A) 6, 1

B) 5, 7

C) 3, 2

D) 1, 5



2^3 = 8

8/8

2*3 = 6

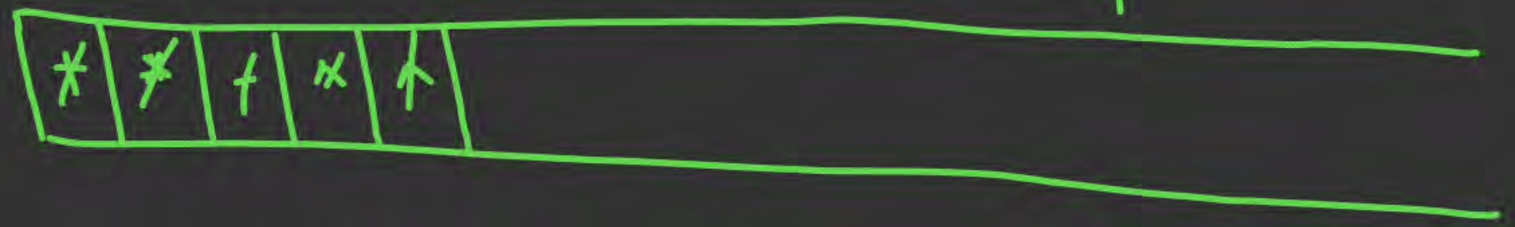
Q) Assume that the operators +, -, * are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, *, +, -. The postfix expression corresponding to the infix expression a + b * c - d ^ e ^ f is

A) abc * + def ^ ^ -

B) abc * + de ^ f ^ -

C) ab + c * d - e ^ f ^

D) - + a * bc ^ ^ def



abc * + def ^ ^ -

Q) The result evaluating the postfix expression

10 5 + 60 6 / * 8 -

1 1

is

A) 284

B) 213

C) 142

D) 71

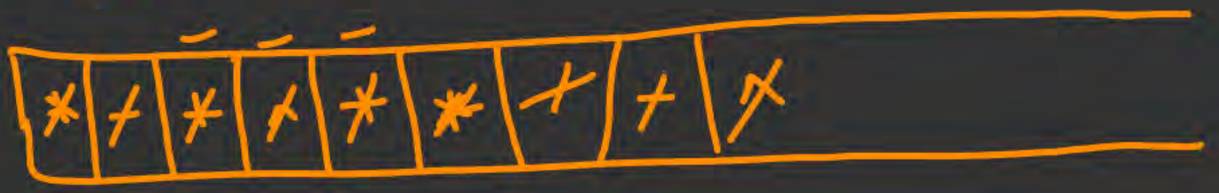
10+5=15 60/6 15*10=150 150-8=142

Que) Consider an infix expression and precedence, associativity as given below. The postfix expression is

X : a * b - c + d ^ e + f * g / h - i ^ j ;

(- : more, - : less)

Operator	Priority	Associativity
^	1	Right to Left
/	2	Left to Right
+	3	Right to Left
*	4	Right to Left
-	5	Left to Right



ab*cde^f++gh/*-ij^-

Ques) Consider precedence and anociativity as given below. convert given Infix expression to Postfix expression

Priority (High to low)	Anociativity
*	R to L
/, +	L to R
^, -	R to L

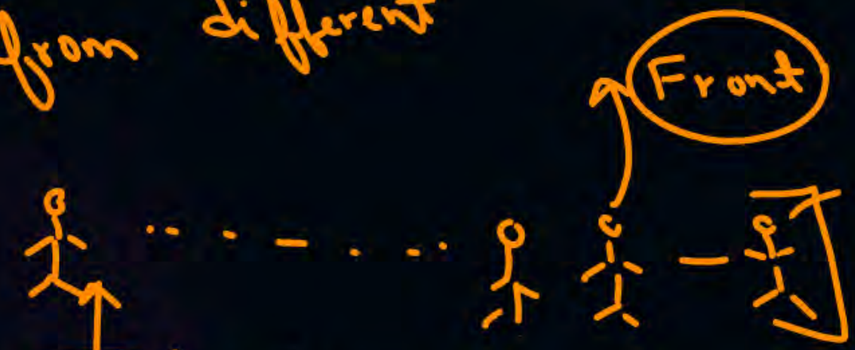
X : 3 + 4 ^ 5 * 6 ^ 2 / 4 - (3 ^ 1) + 7 , * 2

3 4 + 5 6 * 2 4 / 3 1 ^ 7 2 * + - ^ ^

QUEUE

It is a linear data structure in which insertion and deletion may be performed at different end or same end.

If insertion and deletion is from different end, then it means it follows First In First Out (FIFO)



Tykes of Queues

- Simple Queue
- Circular Queue

} FIFO (Front - Deletion)

- Double Ended Queue } Insertion & Deletion can be from same end or different end.
- Priority Queue } It works on the basis of priority not only on arrival time.

In Queue Data Structure Enqueue means inserting the element from the rear end into the queue.
Dequeue means deleting the element from the front of the Queue.

Simple Queue : Let's say we have a Queue(Q) with maxsize = 5

Enqueue(value, Q) : Insert value into Q from the rear end and return nothing

Dequeue(Q) : Delete the value at the front end of the queue & return the deleted value.

```
q = Dequeue(Q)
print(q)
```

```
q = Enqueue(value, Q)
print(q)
```

None

rear = None
front = None

Enqueue(10, Q) ⇒ rear = 0, front = 0
Q[0] = 10

Enqueue(20, Q) ⇒ rear = rear+1 = 1,
front = 0
Q[1] = 20

Enqueue(30, Q) ⇒ rear = rear+1 = 2
front = 0
Q[2] = 30

Enqueue(40, Q) ⇒ rear = rear+1 = 3
front = 0
Q[3] = 40

Dequeue(Q) ⇒ Delete Q[0]
Delete 10
front = front+1 = 1
rear = 3

Enqueue(50, Q) ⇒ rear = rear+1 = 4
Q[4] = 50
front = 1

Dequeue(Q) ⇒ Delete 20
front = front+1 = 2

Enqueue(60, Q) ⇒ Overflow

Dequeue(Q) ⇒ front = 3
rear = 4

Dequeue(Q) ⇒ front = 4

Dequeue(Q) = front = None
rear = None

if (front =

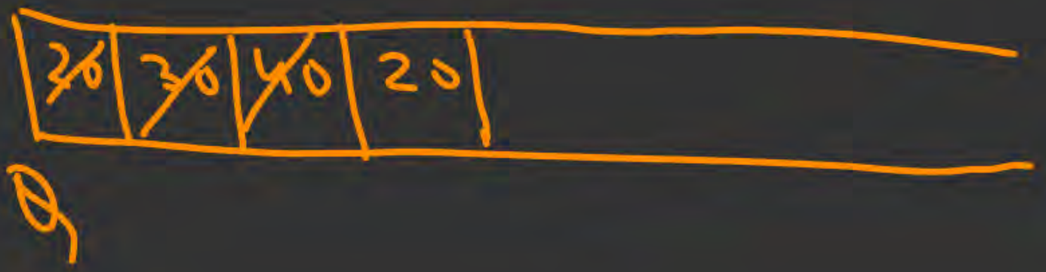
Q_un) Consider an empty stack 'S' and an empty simple queue 'Q'. Perform below operations in the same order.

① Push(10, S)
② Enqueue(20, Q)
③ Push(30, S)
④ Enqueue(Peek(S), Q)
⑤ Push(Dequeue(Q), S)
⑥ Enqueue(40, Q)
⑦ a = Dequeue(Q)

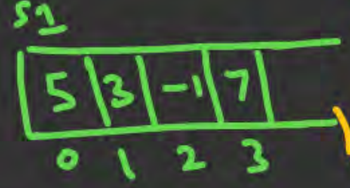
a = 30

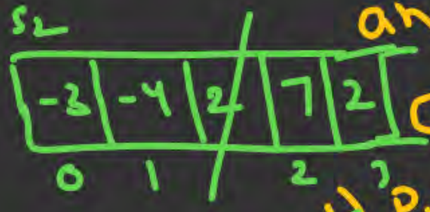
⑧ b = Pop(S) b = 20
⑨ Push(a, S)
⑩ Enqueue(b, Q)
⑪ c = Peek(S) + Dequeue(Q)
30 + 40

C = 70

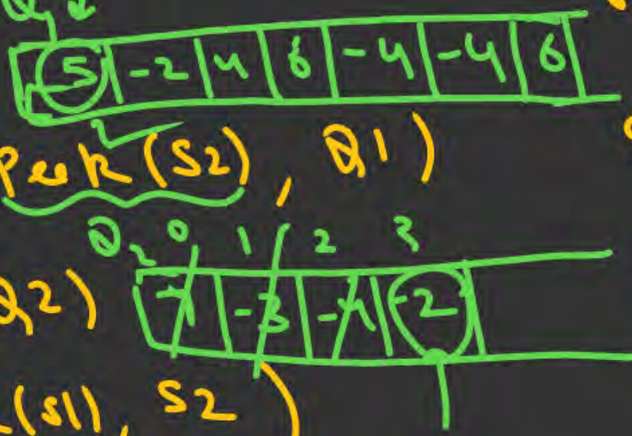


Ques) Consider 2 stacks S1, S2 with elements $\{-1, 3, 5\}$ and $\{2, -4, -3\}$ respectively from Top to bottom. Consider 2 Queues Q_1, Q_2 with elements $\{5, -2, 4, 6\}$ and $\{7, -3, -4, -2\}$ respectively in the same order. Consider below operations:





1) Push (Dequeue(Q_2), S1)
2) POP(S2)
3) Enqueue (Peek(S2), Q1)
4) Dequeue(Q2)
5) Push(Peek(S1), S2)
6) Enqueue (Dequeue(Q2), Q1)



7) x = TOP(S2) + front(Q1) = 2 + 0 = 2
8) y = rear(Q2) + TOP(S1) = 3 + 6 = 6
9) Enqueue(y, Q1)
10) Push(x, S2)
11) z = Peek(S1) + Top(S2) + rear(Q2) + front(Q1)
= 13

index, value, value, 3, 3, 0

GO , ISRO , BARC

THANK - YOU