Stack Permutation and Advance Topics

Course on C-Programming & Data Structures: GATE - 2024 & 2025

Vishvadeep Gothi • Lesson 37 • Jan 12, 2023

Data Structure:
Doubts & Stack

By: Vishvadeep Gothi

Hello!

I am Vishvadeep Gothi

I am here because I love to teach

Question GATE-2001

What is the minimum number of stacks of size n required to implement a queue of size n ?

(A) One

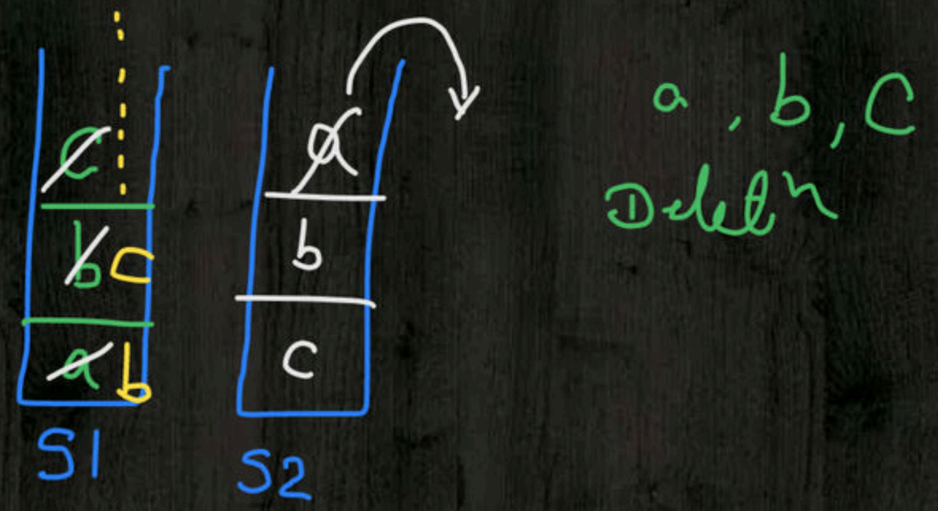
(B) Two

(C) Three

(D) n

I have ⇒ Stack (LIFO)

I want ⇒ Queue (FIFO)



a, b, C
Delete

Implementation of Queue Using Stack

Best method:-

```
Insert(Q)
{
  PUSH (S1)
}
```

```
Delete()
{
```

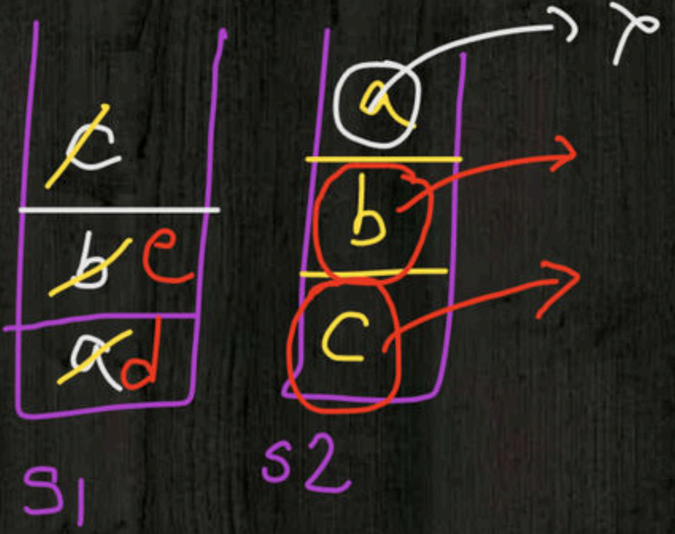
1. if S2 is not empty then
   pop from S2

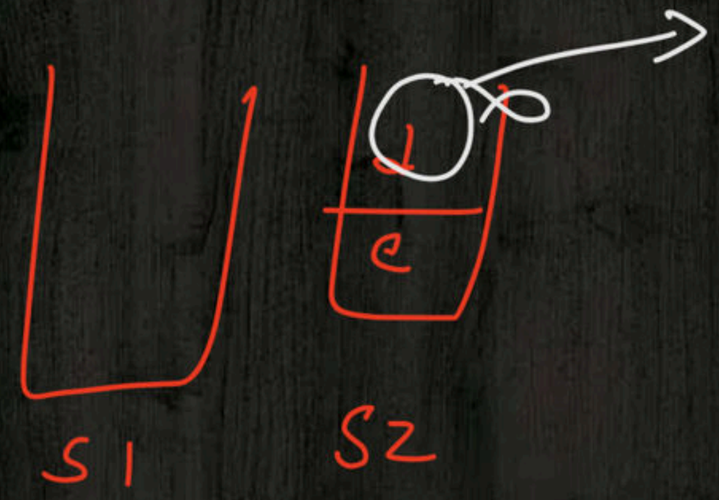
2. if S2 is empty then
   bring all elements from S1 to S2
   and the pop from S2

```
}
```

Implementation of Queue Using Stack

1. Insert (a)
   Insert (b)
   Insert (c)
   Delete ()
   insert (d)
   insert (e)
   Delete ()
   Delete ()
   Delete ()
   Delete ()





Question GATE-2006

An implementation of a queue Q, using two stacks $S1$ and $S2$, is given below:

```
void insert (Q, x) {
    push (S1, x);
}
void delete (Q) {
    if (stack-empty(S2)) then
        if (stack-empty(S1)) then {
            print("Q is empty");
            return;
        }
        else while (!(stack-empty(S1))){
            x=pop(S1);
            push(S2,x);
        }
    x=pop(S2);
}
```

m => insert

n => delete

PUSH => x

POP => y

let n *insert* and $m(\leq n)$ *delete* operations be performed in an arbitrary order on an empty queue Q. Let x and y be the number of *push* and *pop* operations performed respectively in the process. Which one of the following is true for all m and n?

A. $n+m \leq x < 2n$ and $2m \leq y \leq n+m$

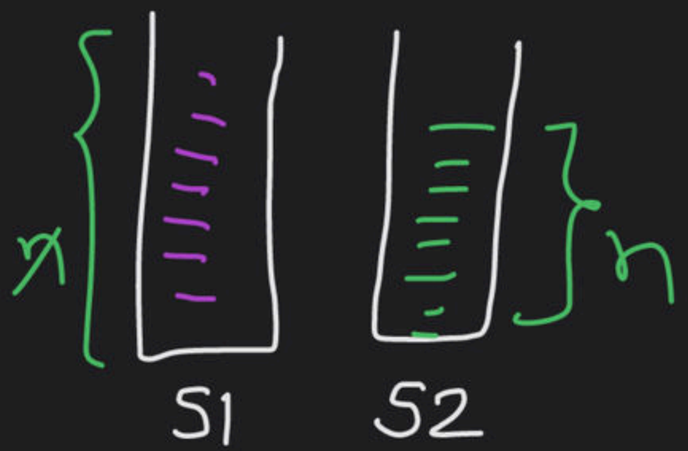
B. $n+m \leq x < 2n$ and $2m \leq y \leq 2n$

C. $2m \leq x < 2n$ and $2m \leq y \leq n+m$

D. $2m \leq x < 2n$ and $2m \leq y \leq 2n$

Worst Case :-

Insert all n elements then delete m elements



Insertn :- PUSH = n ✓

Deletion :- POP = n

PUSH = n ✓

POP = m

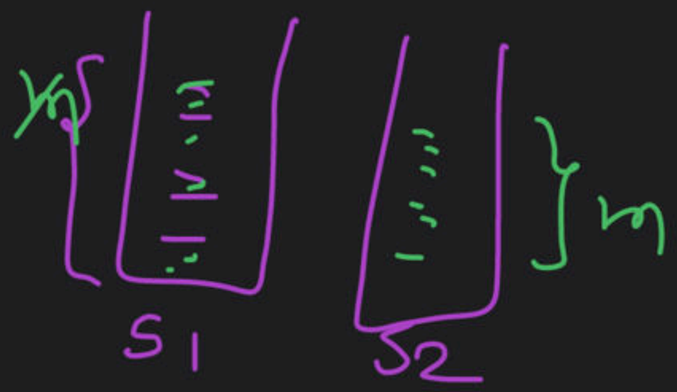
Total

PUSH : 2n

POP = n+m

Best case :-

Insert m elements first, delete all of them, then insert remaining n-m elements



Total PUSH = n + m

POP = 2m

Insertion :-

PUSH = m + n - m = n

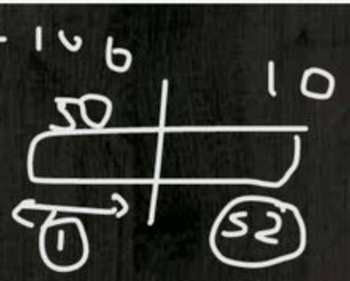
Deletion :-

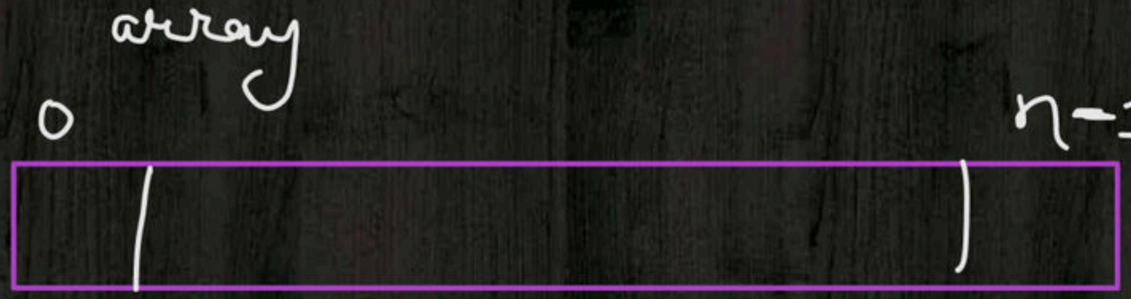
POP = m

PUSH = m

POP = m

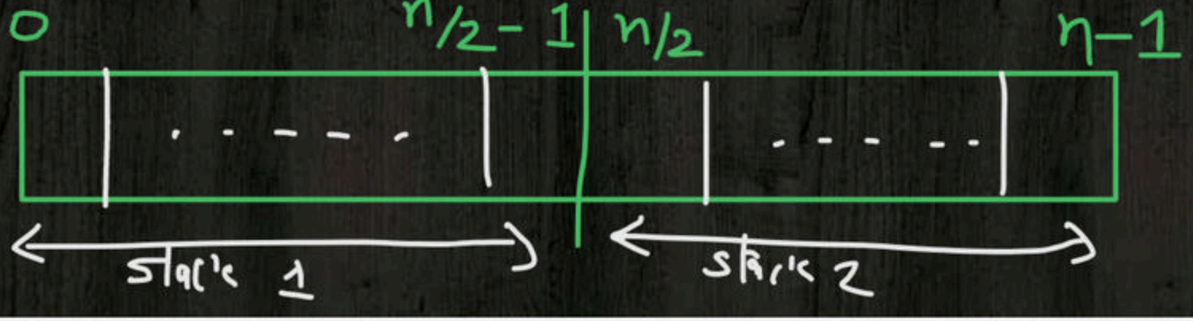
Multiple Stacks in Single Array





Method 1 :- → space utilizat^n of array is not proper

Divide array in 2 equal halves & implement one stack in each half



stack 1 :-

top1 = -1

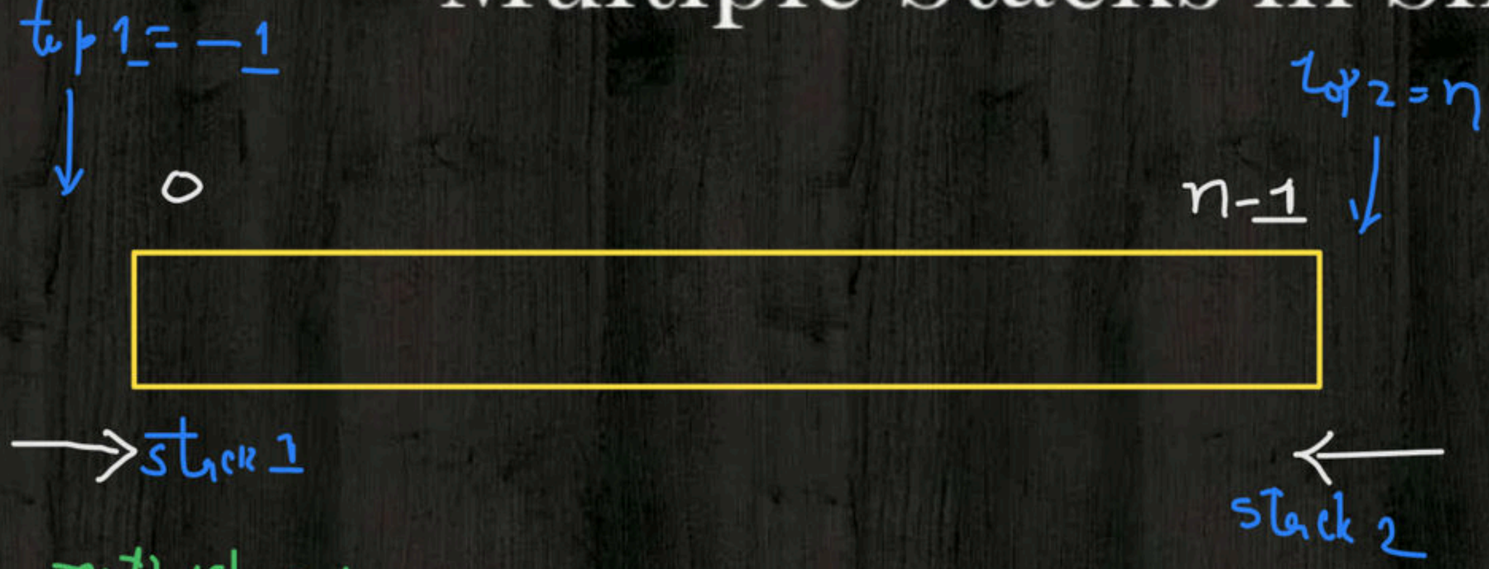
maxtop1 = n/2 - 1

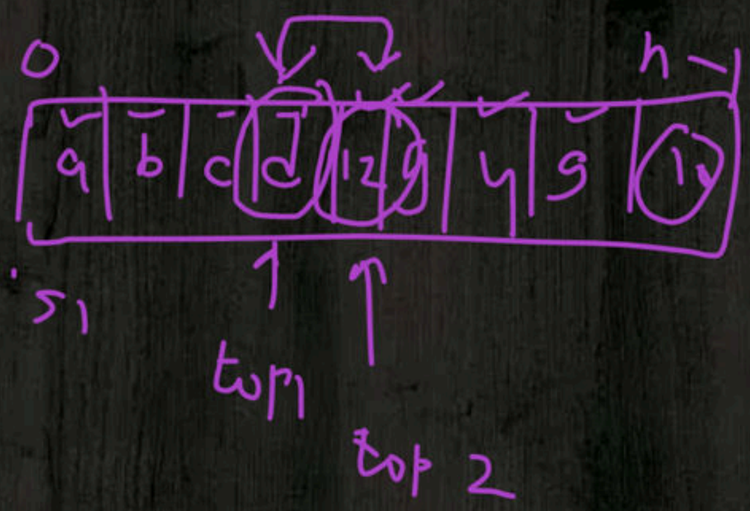
stack 2 :-

top2 = n/2 - 1

max top2 = n - 1

Multiple Stacks in Single Array





Method 2:-

both stacks grows from 2 diff ends.
and can perform insertn
until there is a free space available

overflow condition :-

top 1 == top 2 - 1

or

top 2 == top 1 + 1

---

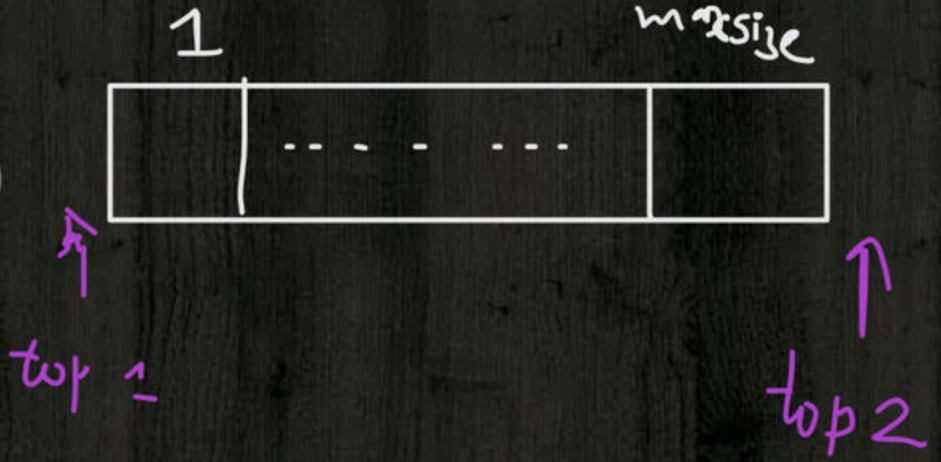
underflow condition :-

top 1 == -1

top 2 == n

Question GATE-2004

A single array A[1..MAXSIZE] is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top1 and top2 (top1< top 2) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for “stack full” is



(A) (top1 = MAXSIZE/2) and (top2 = MAXSIZE/2 + 1)

(B) top1 + top2 = MAXSIZE

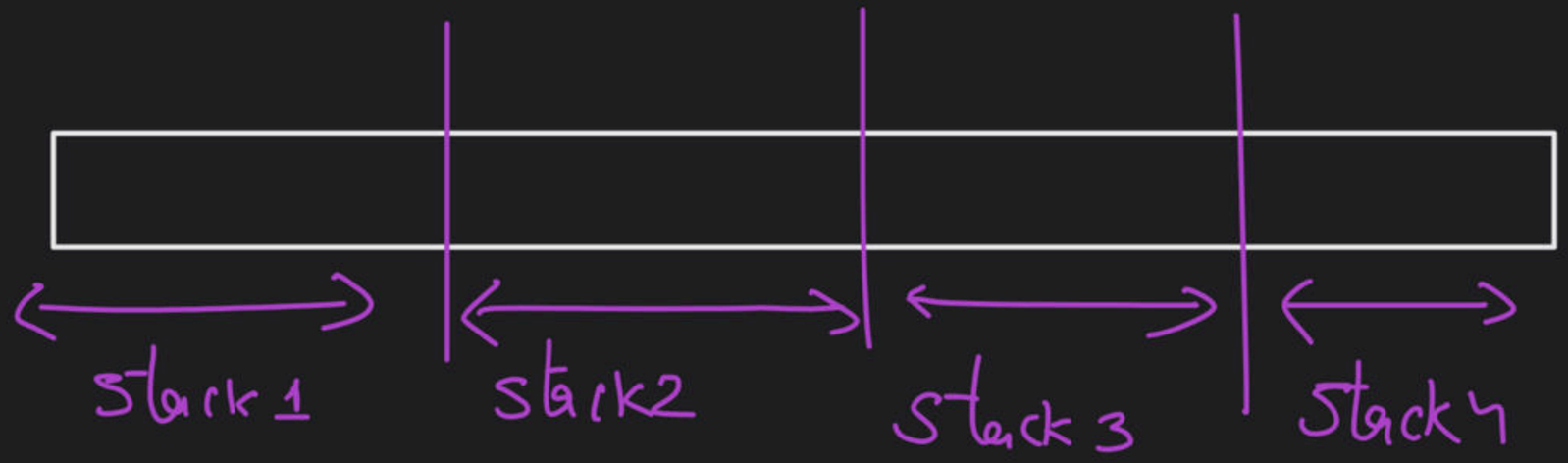
(C) (top1= MAXSIZE/2) or (top2 = MAXSIZE)

(D) top1= top2 -1

if more than 2 stacks implemented in single array

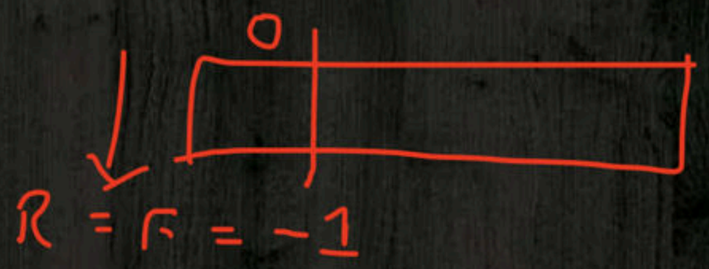
⇓

soln ⇒ give equal space to each stack.

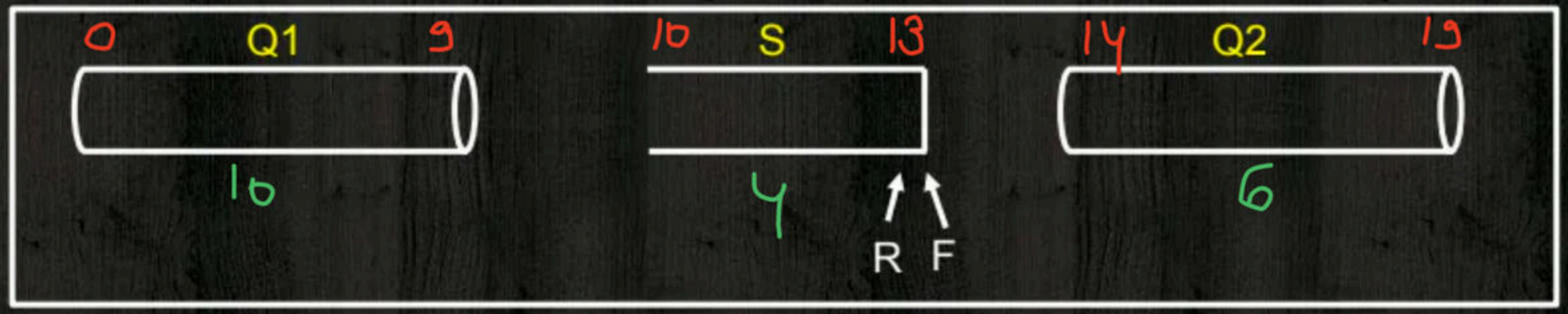


DPP

Question 1



Two queues Q1, Q2 and a stack S are implemented using a single array of size 20. Array has indexes 0 to 19 and has a base address $(504)_{10}$. The stack S and Queue Q2 are initially empty. The Q1, Q2 and S can be visualized as follows:



Size of Q1 = 10, Q2 = 6

The queue Q contains values: 5,9,3,4,0,7,9,4,0,6 from front to rear in that order. The array is of integers where each element takes 2 locations in memory. Consider the following program fragment executed on this array (on Q1, Q2 and S).

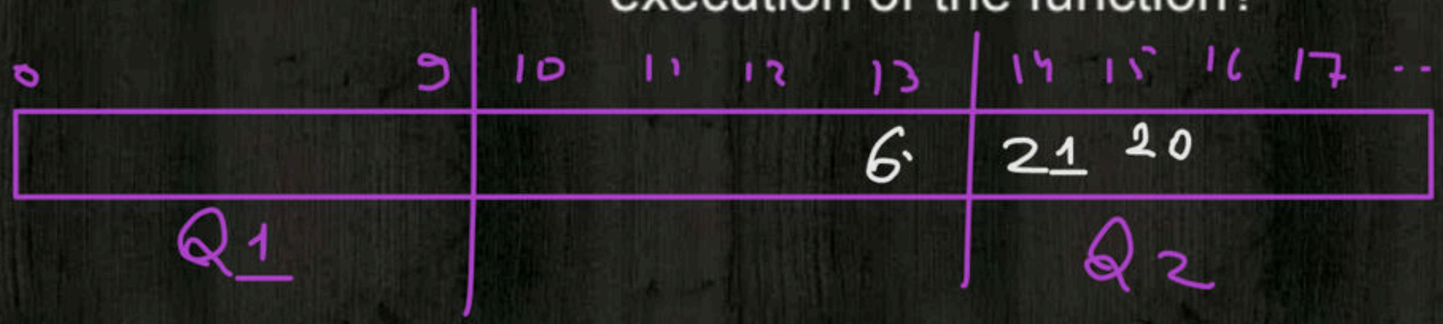
Question 1 cont..

Q1 => 5, 9, 3, 1, 0, 7, 3, 4, 6

```
Void doSomething()
{
  int x, y, z;
  while(!IsEmptyQueue(Q1))
  {
    x = dequeue(Q1);
    if(x ! = 0)
        PUSH(S, x);
    else
    {
        z=0;
        while(! IsEmptyStack(S))
                {
                    y=POP(S);
                    z+=y;
                }
    } } }
```

enQueue(Q2, z);

What are the addresses of elements 21 and 6 after execution of the function?



x = 7 3 1 0 6

y = 1 3 7

z = 0 1 13 20

loc(A[14]) = 504 + 2 * (14) = 532

for 21

loc(A[13]) = 504 + 2 * 13 = 530

for 6

Question 2

Let **S** be a stack of size $n \geq 1$. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that Push and Pop operations take X seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation. For $m \geq 1$, define the stack-life of m as the time elapsed from the end of $Push(m)$ to the start of the pop operation that removes m from **S**. The average stack-life of an element of this stack is

A. $n(X+Y)$ B. $3Y+2X$ C. $n(X+Y)-X$ D. $Y+2X$

Question 3

Let Q denote a queue containing sixteen numbers and S be an empty stack. $Head(Q)$ returns the element at the head of the queue Q without removing it from Q. Similarly $Top(S)$ returns the element at the top of S without removing it from S. Consider the algorithm given below.

```
while Q is not Empty do
  if S is Empty OR Top(S) ≤ Head (Q) then
     x:= Dequeue (Q);
     Push (S, x);
  else
     x:= Pop(S);
     Enqueue (Q, x);
  end
 end
```

The maximum possible number of iterations of the **while** loop in the algorithm is _____.

Question 4

Consider an array of size n to implement m number of stacks (numbered 0 to m–1). The empty stack has following specifications:

for $0 \leq i < m$

$$\text{top}[i] = \text{bottom}[i] = i * \left\lfloor \frac{n}{m} \right\rfloor$$

for $i = m$

$$\text{bottom}[i] = n - 1$$

$\frac{n}{m}$ => integer

Where bottom is constant index and top is variable (moving) index.

Question 4 continue

Fill in the blank:

1.

```
void push (int i, char item)
{
   if(_____)
            {
               printf("Overflow");
            }
   else
            {
               stack[++top [i]] = item;
            }
}
```

Question 4 continue

Fill in the blank:

2.

```
void pop(int i)
{
   if(_____)
            {
               printf("Underflow");
            }
   else
            {
               stack[top[i]--];
            }
}
```

Happy Learning