Page No.:
Date: / /

(Lecture - 37)

Stack - Hard

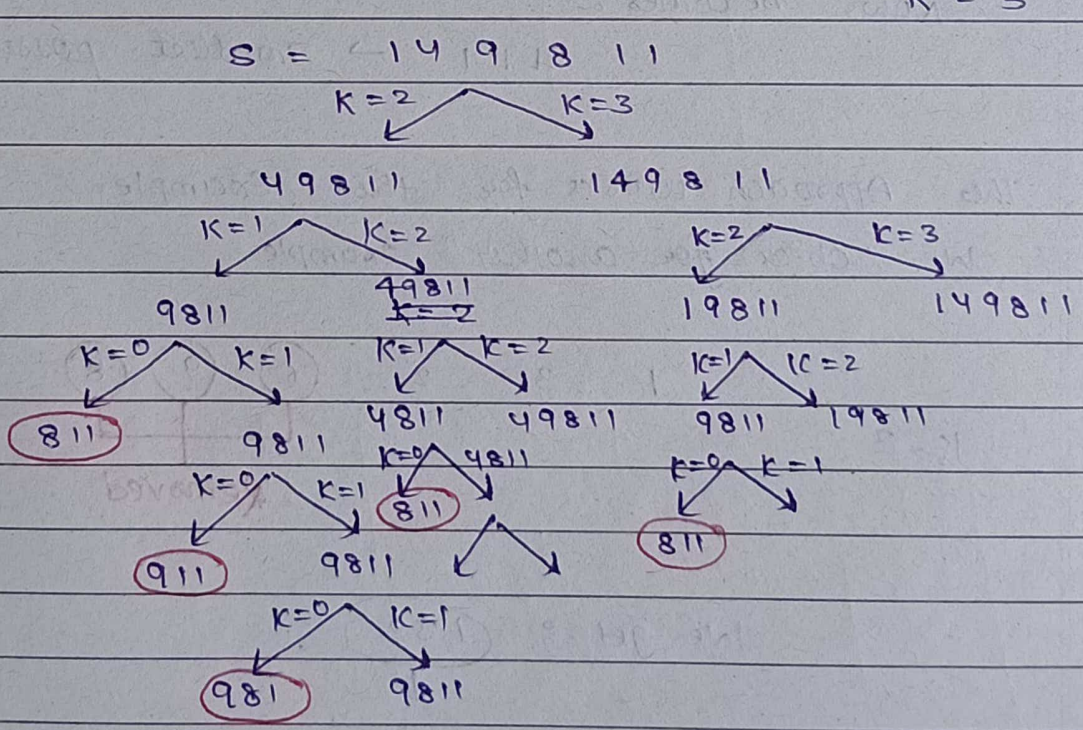
Problem 1 : Remove k digits

Given : A Non negative Integer represented as string

Task : Remove k digits from number, so, new number becomes smallest possible.

give num does not contain any leading zero.

Approach 1 : (By recursion)



Compare all the outcome where k = 0, and find the smallest among them and return

Teacher's Signature.....

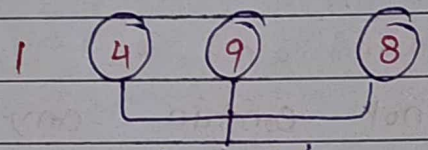
Time Complexity : $O(2^n)$

But we have to solve in $O(n)$.
We find another Approach.

Approach 2 :

1 4 9 8 1 1 (k = 3)

Remove the k digit which is largest



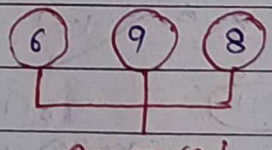
Removed
3 digit

Now, becomes :

1 1 1 → smallest possible

This Approach works for this Example.
We check for another example.

k = 3



Removed

We get : 132

↓

is it smallest possible ? No.
because 126 is smaller than 132

This Approach fails.

Teacher's Signature.....

Page No.:
Date: / /

Approach 3:

1 3 2 6 9 8

k = 3

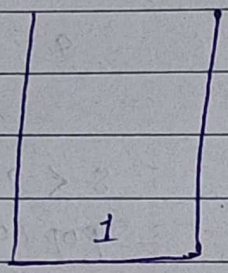
Make a stack

1 3 2 6 9 8

↑

Stack empty

push(arr[i]);



1 3 2 6 9 8

↑

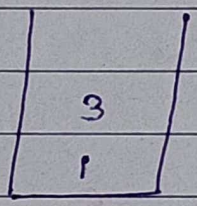
if (arr[i] < top())

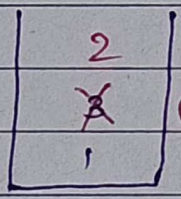
pop(top());

k--;

else if (arr[i] > top())

push(arr[i]);





k = 2

1 3 2 6 9 8

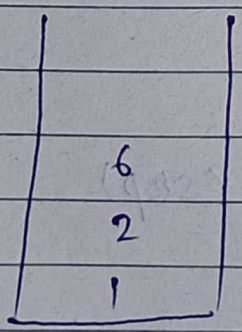
↑

2 < 3 ← top

pop() → 3

K--;

push(2)

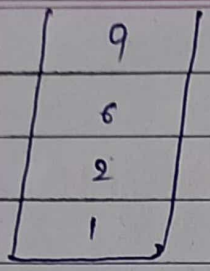


1 3 2 6 9 8

↑

6 > 2

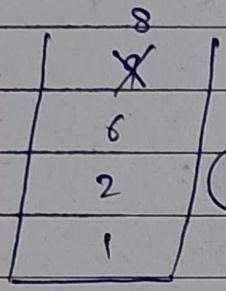
Teacher's Signature.....



1 3 2 6 9 8

9 > 6

push (9)



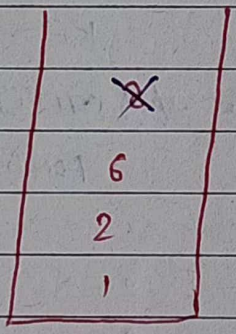
1 3 2 6 9 8

8 < 9

pop (9)

push (8)

At last, we have

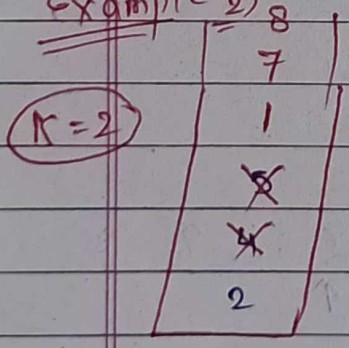


K = 1 → remove from top

pop (8)

Ans = (1 2 6)

Example 2)



2 4 5 1 7 8

push(2)

4 > 2
push(4)

5 > 4
push(5)

1 < 5
pop(5) K = 1

1 < 4
pop(4) K = 0 (stop)

push

push

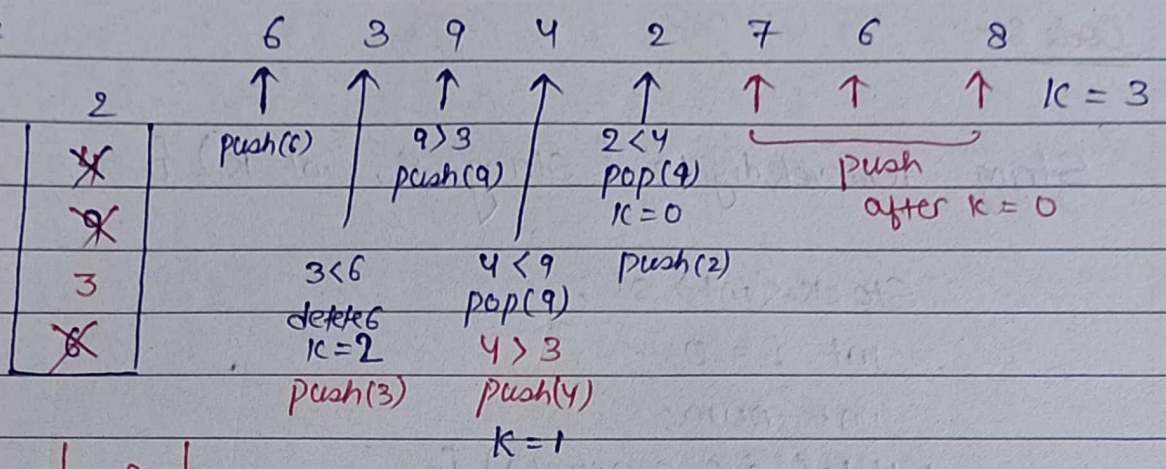
8 7 1 2

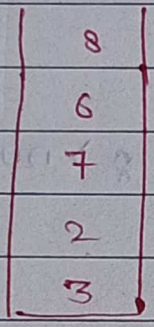
→ Reverse

Ans 2178.

Teacher's Signature.....

Example



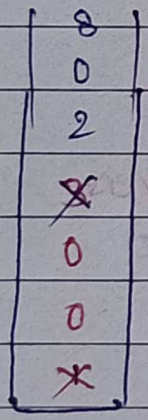


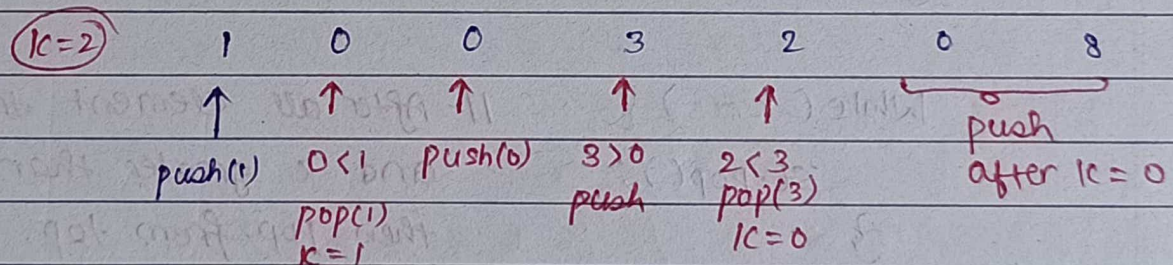
→ 8 6 7 2 3 → Reverse = Ans

Ans = 3 2 7 6 8

Example

1 0 0 3 2 0 8





8 0 2 0 0 → rev → 0 0 2 0 8

Ans : 2 0 8.

Teacher's Signature.....

Page No:
Date: / /

Code :-

```
String removeKdigits ( string S, int K) {

    Stack <int> st;
     int i = 0;
     int num;
    While( i < S.size()) {
        num = S[i] - '0';
      While(st.size() && st.pop() > num && K){
                st.pop();
                 K--;
            }
           st.push(num);
           i++;
    }

  While( K--) {          // After all element traverse
      st.pop();          and K greater than 0
    }                    then pop from top.

  String ans;
  char c;                // stack से बचा
  While ( st.size()) {   हुआ character
      c = '0' + st.top(); निकाल लो।
       st.pop();
      ans += c;
  }
```

Teacher's Signature.....

```
        i = ans.size() - 1;
    while (i >= 0 && ans[i] == '0'){
            ans.pop_back();  // अगर Number के पीछे में zero
            i--;             // है तो उसको Remove.
        }

    reverse(ans.begin(), ans.end());

    if (ans.size() == 0)
        return "0";

    return ans;
}
```

Problem : 2 clumsy Factorial (LeetCode)

factorial : $6! = 6 * 5 * 4 * 3 * 2 * 1$

But in clumsy factorial :

$6! = 6 * 5 / 4 + 3 - 2 * 1$

priority of * & / > priority of + & -

Example :

$10! = 10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1$

first step : We perform only Multiplication & division.

Teacher's Signature.....

Page No.:
Date: / /

$10! = 90/8 + 7 - 6 * 5/4 + 3 - 2 * 1$

$= 11 + 7 - 30/4 + 3 - 2 * 1$

$= 11 + 7 - 7 + 3 - 2$

Now, we perform addition & substration

$= 18 - 7 + 3 - 2$

$= 11 + 3 - 2$

$= 14 - 2$

$= 12$

→ whenever we use stack, we use stack as a memory.

→ stack remembered the previous task that he had done.

Solving using stack:

$10! = 10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1$

	*	/	+	-
	↓	↓	↓	↓
i =	0	1	2	3

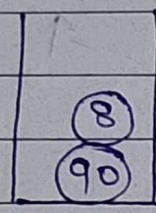
Teacher's Signature.....

→ 

10 9 8 7 6 5 4 3 2 1

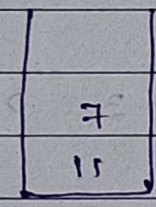
push push

pop both → 9 * 10 = 90 (store this in stack.

→ 

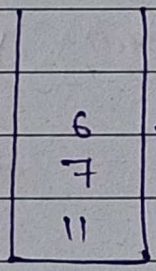
10 9 8 7 6 5 4 3 2 1

pop both → 90 / 8 = 11 (store in stack

→ 

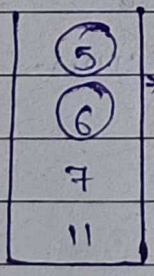
10 9 8 7 6 5 4 3 2 1

We can perform addition, we only perform here Multiplication & division

→ 

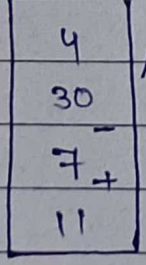
10 9 8 7 6 5 4 3 2 1

can't perform substraction

→ 

10 9 8 7 6 5 4 3 2 1

pop both → 5 x 6 = 30 push

→ 

10 9 8 7 6 5 4 3 2 1

poth both → 30 / 4 = 7 push.

Teacher's Signature.....

3	+
7	-
7	+
11	

10 9 8 7 6 5 4 3 2 1 ↑

Don't Do Add.

2	-
3	
7	
7	
11	

10 9 8 7 6 5 4 3 2 ↑ 1

Don't subtract.

1	*
2	
3	
7	
7	
11	

10 9 8 7 6 5 4 3 2 1 ↑

Pop both → 2 * 1 = 2 push.

2	-
3	+
7	-
7	+
11	

Now loop ends for Multiplication & Division.

No, We pop() one by one element and copy to another stack.

2	
3	
7	
7	
11	

→ Copy in New Stack

11
7
7
3
2

Teacher's Signature.....

Page No.:
Date: / /

pop 2 element & add.

11 + 7 = 18

18
7
3
2

pop 2 element & subtract

18 - 7 = 11

11
3
2

pop 2 element & add

11 + 3 = 14

14
2

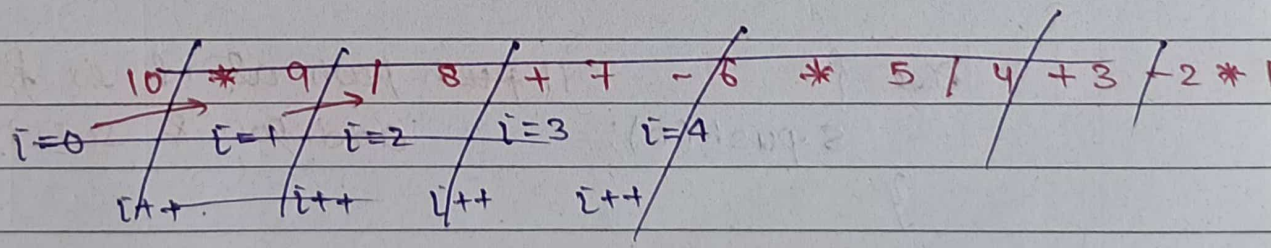
pop & Subtract

14 - 2 = (12)

↳ Required answer.

How to check which operation performed

	*	/	+	-
i =	0	1	2	3



10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1

i=0 i=1 i=2 i=3 (i=4)

i++ i++ i++ i++

↓

This is Not available.

Here we use Modulo.

[i = (i + 1) % 4]

Teacher's Signature.....

Code :

```
Stack<int> s;
int num, int i=0;
   s.push(n);   // if n=10, then 10 goes to
   n--;         stack and loop start with 9
while(n){
// is while loop me sirf * & / karenge.
      if(i==0){  // 0 for Multiply
      num = s.top();
       s.pop();
       s.push(num * n);
      }

else if(i==1){          // for divide.
      num = s.top();
       s.pop();
       s.push(num / n);
      }

else {                  // ~~push~~ push as it is for
      s.push(n);        addition & substraction.
    }

    i = (i+1) % 4;      // i value changes (0,1,2,3)
     n--;
};

// Till Now Multiplication & division done.
```

Teacher's Signature.....

// Now, Copy in another stack

```
        stack <int> ans;
        while (s.size()) {
            ans.push(s.top());
            s.pop();
        };

    int sum = ans.top();
    ans.pop();
    bool flag = 0;
    while (ans.size()){

        if (flag == 0)
            sum += ans.top();
        else
            sum -= ans.top();
        ans.pop();
        flag = !flag;
    }

    return sum;
};
```

This code gives Time limit Exceed

We have to change some of our Approaches

Teacher's Signature.....

~~Problem 3: Max. Rectangle in of Histogram~~

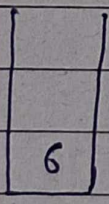
When Multiply divide come we can Multiply & divide then store.

In addition, or substraction We put here (-n) for substraction so, we have perform only addition and no, new stack made.

Ex:

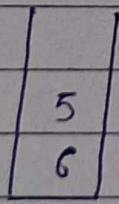
6! = 6 * 5 / 4 + 3 - 2 * 1

↑
i=0

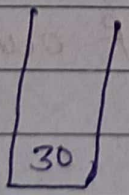


6 * 5 / 4 + 3 - 2 * 1

↑
i=0



= 30



Teacher's Signature.....

6 5 4 3 2 1

↑

i=1

7

30/4 = 7

6 5 4 3 2 1

↑

i=2

3
7

6 5 4 3 2 1

↑

i=3

put (-n) in stack

-2
3
7

6 5 4 3 2 1

↑

i=1

-2
3
7

-2 * 1 = -2

After this, we have to only add.

-2 + 3 + 7

= (8)

Ans

Teacher's Signature.....

Code :

```
stack <int> s;
 s.push(n);
 n--;

int index = 0;
while(n > 0){
    if(index % 4 == 0){
        int num = s.top();
        s.pop();
        s.push(num * n);
    }

else if(index % 4 == 1){
        int num = s.top();
        s.pop();
        s.push(num / n);
    }

else if(index % 4 == 2){
        s.push(n);
    }

else{
        s.push(-n);
    }

    index++;
    n--;
}
```

Teacher's Signature.....

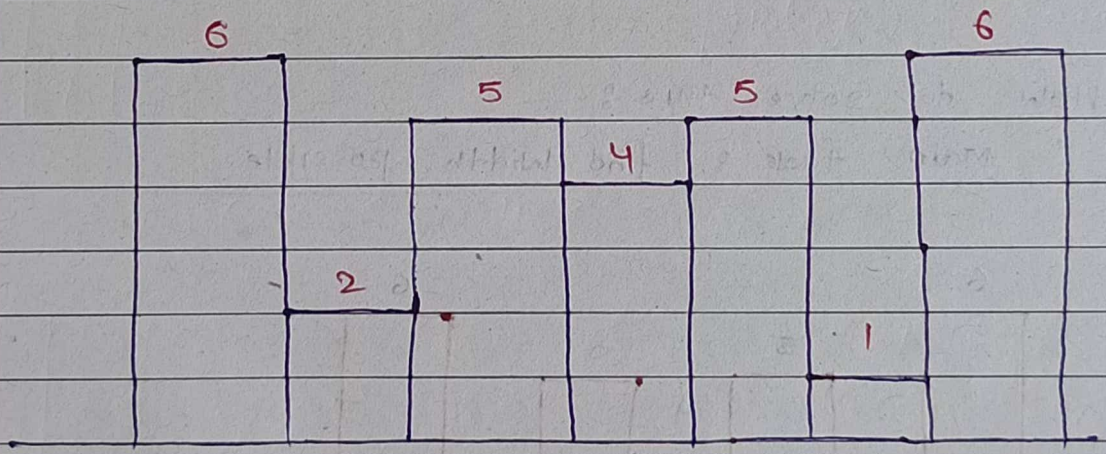
```
int sum = 0;
While (S.size()){
        sum += s.top();
        s.pop();
}

return ~~ans~~ sum;
}
```

Problem 3: Maximum Rectangle area in a Histogram :

largest rectangular are possible

N = 7

arr[] = {6, 2, 5, 4, 5, 1, 6}



Approach 1:

Go and check every every number, how much it can acquire.

for 6 : height = 6 area = 6

width = 1 Max = 6

for 2

height = 2 area = 10

Width possible = 5 Max = 10

for 5

height = 5 area = 5

Width possible = 1 Max = 10

For 4

height = 4 area = 12

Width possible = 3 Max = 12

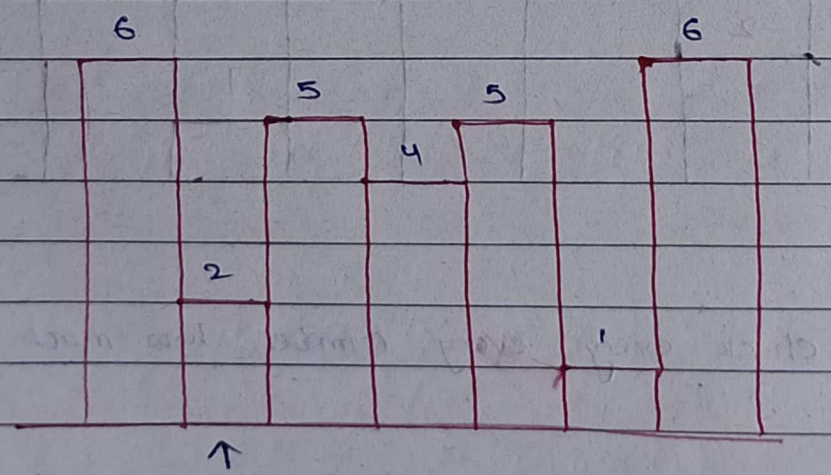
For 5

height = 5 area = 5

Width possible = 1 Max = 12

How to solve this :

Main task : find Width possible.



For 2

How to find Width possible.

Teacher's Signature.....

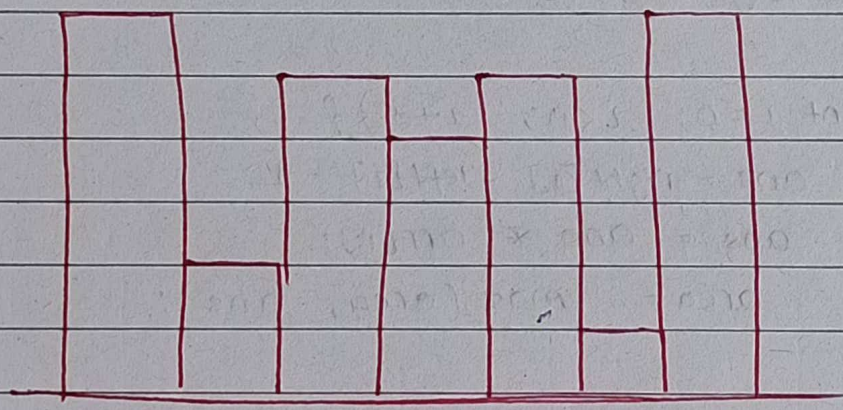
We know that,

Width possible = 5

It can find next minm in both direction, in right and left.

- next smallest right
- next smallest left

We can Find for all and then check :



Height	6	2	5	4	5	1	6
Index	0	1	2	3	4	5	6
Right	1	5	3	1	1	7	7
left	-1	-1	12	1	3	-1	5

> This is index.

Maximum possible

breadth = Right - left - 1;

Area = height[i] * breadth.

Teacher's Signature.....

Code:

```
long long getMaxArea ( log long arr[], int n ){

    int * right  = new int [n];
    int * left   =  new int [n];

    push_right (arr, n, right);
    push_left (arr, n, left);

    long long area = 0;
    long long ans;

    for(int i=0;  i<n;  i++){
        ans = right[i] - left[i] - 1;
        ans = ans * arr[i];
        area =  max (area, ans);
    }
    return area;
}

Void push_left ( long long arr[], int n, int left[]){
        stack<int> s;
        int i = n-1;
        While ( i >= 0) {
            if ( s.size() == 0){
                s.push (i);
            }
            else {
                if ( arr[i] > arr [s.top()]){
```

```
            s.push(i);
        }
        else{
            while(s.size() && arr[i] < arr[s.top()]){
                left[s.top()] = i;
                s.pop();
            }
            s.push(i);
        }
    }
    i--;
}
while (s.size()){
    left[s.top()] = -1;
    s.pop();
}
}

void push_right (long long arr[], int n, int right[]){
    stack<int> s;
    int i = 0;
    while (i<n){
        if (s.size()==0)
            s.push(i)
        else{
            if (arr[i] >=arr[s.top()])
            ~~s.to~~ s.push(i);
            else{
                while(s.size() && arr[i] < arr[s.top()]){
                    right[s.top()] = i;
                    s.pop();
                }
```

Teacher's Signature.....

```
            s.push(i);
        }
    }
    i++;
}
while(s.size()){
    right[s.top()] = n;
    s.pop();
}
}
```

Problem 4 : Max Rectangle :

0	1	1	0
1	1	1	1
1	1	1	1
1	1	0	0

find the Maxm Rectangle of 1's.

We will check here row by row and find maxm area.

Row 1 :

	0	1	1	0

heights : 0 1 1 0

Max = 2 Area = 2

Teacher's Signature.....

Row 2 :

	0	1	1	0
	1	1	1	0 1
Heights :	1	2	2	0 1

Area = 4 Max = 4

Row 3 :

	0	1	1	0
	1	1	1	1
	1	1	1	1
Heights :	2	3	3	2

Area = 8 Max = 8

Row 4 :

	0	1	1	0
	1	1	1	1
	1	1	1	1
	1	1	0	0
Hight =	3	4	0	0

zero because in last Row is 0 so, ~~be~~ without how we make rectangle.

Area = 6 Max = 8

↳ return this

Teacher's Signature.....

Code :

```
int maxArea ( int M[MAX][MAX], int n, int m) {
        int heights[m] = {0};
        int maxArea = 0;

    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            if ( M[i][j] == 0) {
                heights[j] = 0;
            }
            else {
                heights[j] += M[i][j];
            }
        }
        maxArea = max (maxArea, getMaxArea (heights, m));
    }
    return maxArea;
}

int getMaxArea (int heights[], int n ) {
        stack <int> st;
        int maxArea = 0;
        int i = 0;

    while ( i < n ) {
        if (st.empty() || heights[i] >= heights[st.top()]) {
            st.push(i);
            i++;
        } else {
```

Teacher's Signature.....

```
            int tp = st.top();
            st.pop();
            int area = heights[tp] * (st.empty() ? i : (i - st.top() - 1));
            maxArea = max(maxArea, area);
        }
    }

    while (!st.empty()) {
        int tp = st.top();
        st.pop();

        int area = heights[tp] * (st.empty() ? i : (i - st.top() - 1));
        maxArea = max(maxArea, area);
    }
    return maxArea;
}
```

problem 5: The celebrity problem:

Celebrity : known by all but celebrity dont know anyone. in party.

find celebrity present or not in party.

1 → i^{th} know j^{th}

M[i][j] → always 0.

Teacher's Signature.....

N = 3

M[][] =	0	1	2
0	0	1	0
1	0	0	0
2	0	1	0

person 0 → knows 1 can't be celebrity

~~pose~~

person 1 →

person 2 → knows 1 can't be celebrity

person 1 May be celebrity.

check row of 1 → all 0
check colum of 1 → all 1
] person 1 is celebrity.

* How to solve:

celebrity ke row me sare 0
↳ then May be celebrity.

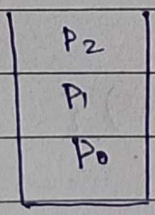
check Column All are 1 except self
↳ then it celebrity.

Teacher's Signature.....

first	second	
0	0	→ Both don't know other → Not celebrity.
0	1	→ second Not celebrity
1	0	→ first Not celebrity
1	1	→ Not celebrity.

	P_0	P_1	P_2
P_0	0	1	0
P_1	0	0	(0)
P_2	0	(1)	0

put all person in stack



POP 2

P_1

0

$P_2 \rightarrow P_1$

$P_1 \rightarrow P_2$

P_2 can't be celebrity

P_1, may be celebrity

push (P_1)

Teacher's Signature.....

P_1
P_0

	P_0	P_1
	1	0

P_0 can't be celebrity

P_1 may be celebrity.

push(P_1)

P_1

check the row of P_1

↳ all zero

check the col-n of P_1

↳ all 1 except self.

code:

```
int celebrity (vector<vector<int>> &M, int n){
        stack<int> s;
    for (int i = 0; i<n; i++){
        s.push(i);
    int first, second;

    while (s.size() > 1){
        first = s.top();
        s.pop();
        second = s.top();
        s.pop;
```

Teacher's Signature.....

```
            if (M[first][second] && !M[second][first]){
                    s.push(second);
            }
            else if (M[second][first] && !M[first][second]){
                    s.push(first);
            }
        }
    }

    if (s.size() == 0){
        return -1;
    }

    int check = s.top();
    int count_row = 0, count_col = 0;

    for(int i = 0; i<n; i++){
        count_col += M[i][check];
        count_row += M[check][i];
    }

    if (count_row == 0 && count_col == n-1){
        return check;
    }
    else{
        return -1;
    }
}
};
```

Teacher's Signature.....