Todays Content:

Problem Description

There are **B** flights labeled from **1** to **B**.

You are given a **2D** array of flight bookings **A**, where **A[i]** represents a booking for flights numbers from **A[i][0]** through **A[i][1]** (inclusive) with **A[i][2]** seats reserved for each flight in the range.

Return an array of length **B**, where each element at index **i** is the total number of seats reserved for flight **i**.

Note : For example if A[i] = {2,4,3} that means in every flight from range 2 to 4, 3 seats are reserved.

Problem Constraints

$1 <= |B| <= 10^5$

$1 <= |A| <= 10^5$

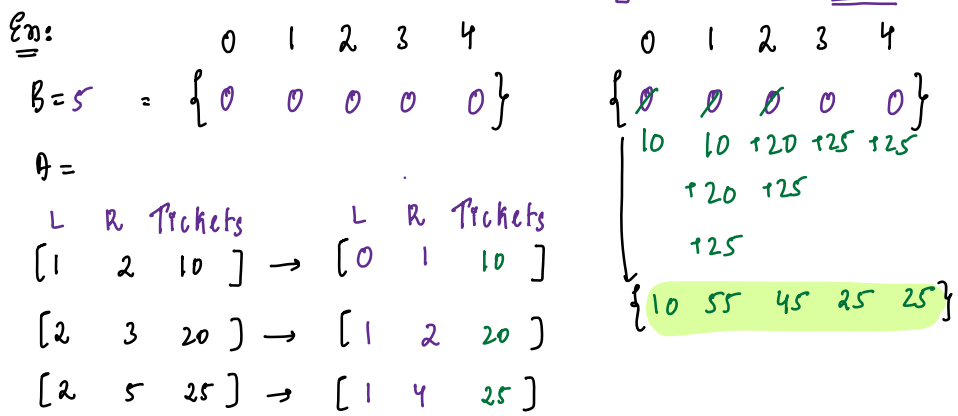
$|A[i]| == 3$

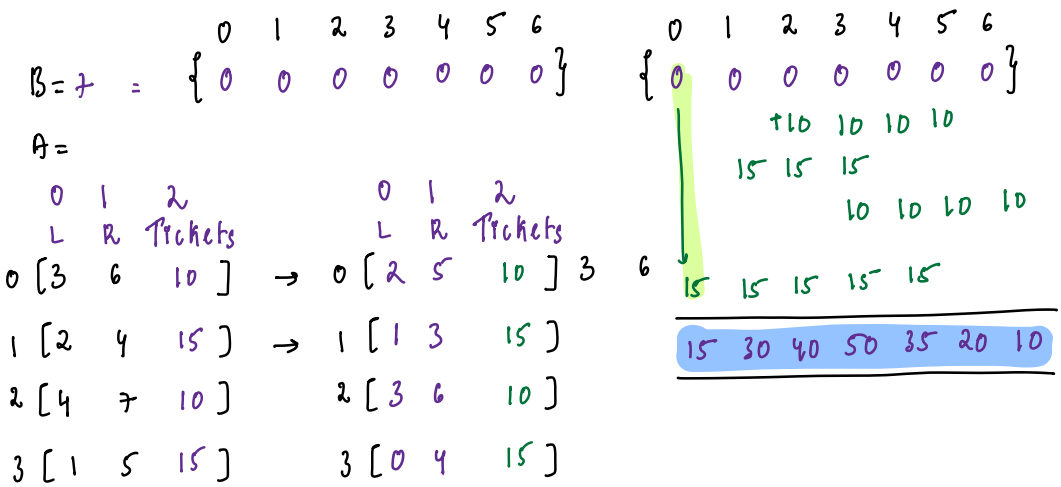
$1 <= A[i][0] <= A[i][1] <= n$

$1 <= A[i][2] <= 10^4$

Note: Flight numbers are given from 1...B

In gener arr[] 0...B-1





B = 7 = { 0 0 0 0 0 0 0 } (indices 0 1 2 3 4 5 6)

A[4][3]

{ 0 0 0 0 0 0 0 }
15 15 10 10 -15 -15 -10
: 15 30 40 50 35 20 10

Idea: cnt[l] += Tickets cnt[R+1] -= Tickets

	L	R	Tickets		L	R	Tickets	
0	3	6	10	→	2	5	10	: cnt[2] += 10 cnt[6] -= 10
1	2	4	15	→	1	3	15	: cnt[1] += 15 cnt[4] -= 15
2	4	7	10	→	3	6	10	: cnt[3] += 10 cnt[7] : no need of Subrat
3	1	5	15	→	0	4	15	: cnt[0] += 15 cnt[5] -= 15

```
int[] FlightBook(int B, int A[][]){
    int cnt[] = new int[B];
    int Q = A.length;
    for(i=0; i<Q; i++){
        int l = A[i][0]; l = l-1;
        int r = A[i][1]; r = r-1;
        int t = A[i][2]
        // Perform Query, add t from index [l...r]
        cnt[l] = cnt[l] + t;
        if(r+1 < B){
            cnt[r+1] = cnt[r+1] - t;
        }
    }
    // last Step: PrefixSum on arr[] & return.
    TODO: Apply psum on cnt[] & return
}
```

Problem Description

Given a rectangular matrix **A** of **N×M** dimension. Return its boundary in clockwise direction.

Problem Constraints

$1 <= N, M <= 10^3$

$1 <= A[i][j] <= 10^9$

Input Format

First argument A is a 2D integer array

Output Format

Return an array of integers denoting the boundary elements of the matrix

Ex1:

mat[4][5]

	0	1	2	3	4
0	2	6	8	4	10
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	16

output:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
boa[14] =	2	6	8	4	10	5	10	16	14	13	12	11	6	1

2N + 2M - 4 = 2*{4} + 2*{5} - 4 = 8 + 10 - 4 = 14

Ex2:

	0	1
0	1	2
1	3	4
2	5	6
3	7	8
4	9	10

output:

2N + 2M - 4 = 2*{5} + 2*{2} - 4 = 10 + 4 - 4 = 10

Ex3: mat[5][1]

	0
0	3
1	6
2	9
3	10
4	14

2N + 2M - 4

2x5 + 2x1 - 4

10 + 2 - 4 = 8

boa[5] : {3 6 9 10 14}

mat[1][5]

Ex4:

	0	1	2	3	4
0	7	2	14	10	8

2N + 2M - 4

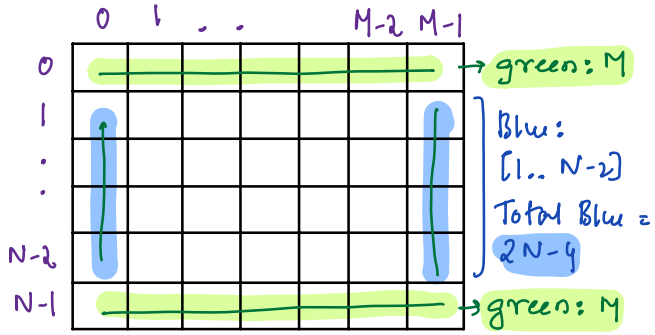
2x1 + 2x5 - 4 = 8

boa[5] = {7 2 14 10 8}

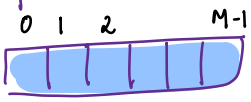
int mat[N][M], N Rows, M Columns:

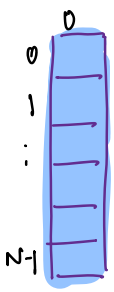
For Each Row: M Columns For Each Column: N Rows

Total ele: 2N + 2M - 4



```
int[] boundary(int mat[][]){
    int N = mat.length;
    int M = mat[0].length;
    if(N==1||M==1){
        if(N==1){// mat size : 1*M
            int bol[M];
            int k=0;
            for(int j=0;j<M;j++){
                bol[k] = mat[0][j];
                k=k+1
            }
            return bol;
        }
        if(M==1){// mat size : N*1
            int bol[N]
            int k=0;
            for(int i=0;i<N;i++){
                bol[k] = mat[i][0]
                k=k+1
            }
            return bol;
        }
    }
    int bod[2*N+2*M-4];
    int k=0;
    //Step1: Iterate on 0th row:
        for(int j=0;j<M;j++){
            bod[k] = mat[0][j]
            k=k+1;
        }
    //Step2: Iterate on M-1st Col
        for(int i=1;i<N-1;i++){
            bod[k] = mat[i][M-1];
            k=k+1;
        }
    //Step3: Iterate on N-1th row:
        for(int j=M-1;j>=0;j--){
            bod[k] = mat[N-1][j]
            k=k+1;
        }
    //Step4: Iterate on 0th Col:
        for(int i=N-2;i>=1;i--){
            bod[k] = mat[i][0];
            k=k+1;
        }
    return bod;
}
```





Problem Description

Construct a binary number having **A** 1's followed by **B** 0's. Return the decimal value of that binary number.
For eg -
A = 3, B = 2 : 11100 → decimal: 16+8+4 = 28
Answer = $(11100)_2$. Return = 28

Problem Constraints

1 <= A + B <= 30

Input Format

The first argument is a single integer A.
The second argument is a single integer B.

Output Format

Return a single integer that is the decimal value of the converted binary number.

Eg:

A = 4 B = 2 : $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ → 1 1 1 1 0 0 → 32+16+8+4 = 60

Note:

$1 << N = 2^N$

$2^0 + 2^1 + 2^2 + \dots + 2^k = 2^{k+1} - 1$

Idea: $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ decimal

A = 5 B = 3 : val = → 1 1 1 1 1 → $2^0 + 2^1 + 2^2 + 2^3 + 2^4 = (2^5 - 1)$ → $(1<<5) - 1$

$[(1<<5) - 1] << 3$ val<<3 = 1 1 1 1 1 0 0 0 → val<<3 → $[(1<<5) - 1] << 3$

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ decimal

A = 2 B = 5 : val = → 1 1 → $2^0 + 2^1 = (2^2 - 1)$ → $(1<<2) - 1$

$[(1<<2) - 1] << 5$ val<<5 = 1 1 0 0 0 0 0 → val<<5 → $[(1<<2) - 1] << 5$

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ decimal

A = 4 B = 3 : val = 1 1 1 1 → $2^0 + 2^1 + 2^2 + 2^3 = (2^4 - 1)$ → $(1<<4) - 1$

1 1 1 1 0 0 0 → val<<3 → $[(1<<4) - 1] << 3$

$[(1<<4) - 1] << 3$

Given A, B ⇒ $[(1<<A) - 1] << B$