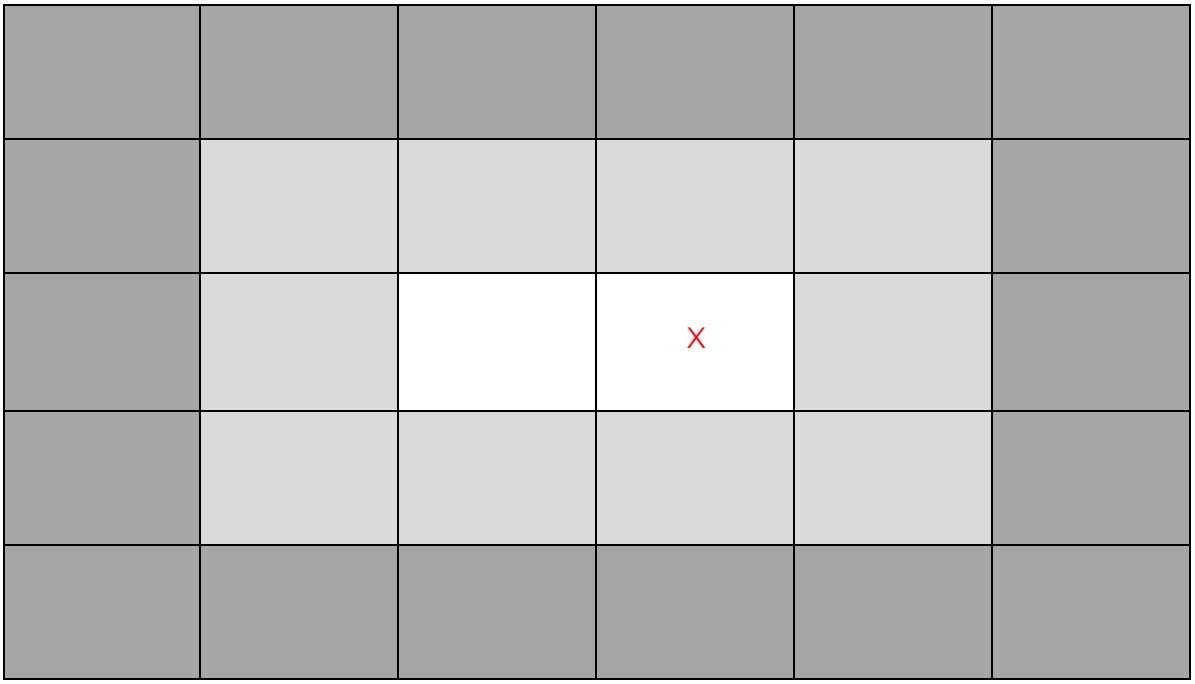
ACM-NIT, Surat
Inception 5.1
Editorial

End cell

Hint: If the last printed cell is (i,j) for matrix (M-2)*(N-2), then the last printed cell for matrix M*N will be (i+1,j+1).



Reduce M and N by 2, while both are greater than 2.

Let **offset** = number of times above operation is performed.

offset=0

```
while(M>2 && N>2) {
    M  = M-2;
    N = N-2;
    offset++;
}
```

offset can be calculated using simple math.

Now there are 4 cases that are possible.

1) N <= M and N=1

2) N <= M and N=2

3) N > M and M=1

4) N > M and M=2

if last cell of either case is (p,q) then final answer will be (p+offset,q+offset)

Michael and sales

There are 2 ways to solve this problem.

(i) By maintaining 10 variable sorted array, which will store the 10 maximum sales we have encountered so far.

We have to update this 10 variable array as we go to the next index. For updating we will one by one compare this new element of input array with all elements of 10 element sorted array from higher to lower. If we find an element (in 10 variable array) smaller than this new element then we will insert this element at this position and discard the smallest element of 10 variable array.

Answer for each index will be summation of all elements in this 10 element array.

(ii) Using priority queue or set. We insert the value of each index in the queue as we move forward.

Answer for each index will be summation of last 10 elements of the set or top 10 elements of priority queue obtained after removing and reinserting 10 elements one by one.

Conflict in Dungeon

- Run a for loop for i from 0 to n-1. For each iteration add one point to Ishan's record if the difference between absolute (A[i%(A.size())] - B[i%(B.size())]) is divisible by only 3 and not 5, viceversa add a point to Krunal's record and in any other case don't add any point to anyone's record.

Parent & Child

- The parent of the n-th node in k-ary tree = floor((n+k-2)/k)
- The m-th child of n-th node in k-ary tree = (n-1)*k + m + 1
- Time Complexity = O(Q)

Maximum Cost

- This problem is the variation of standard LCS(longest common subsequence) problem.
- States:
    - if (a[i - 1] == b[j - 1]) {
        dp[i][j] = dp[i - 1][j - 1] + cost[a[i - 1]];
      }
      else {
        dp[i][j] = max(dp[i][j - 1], dp[i - 1][j]);
      }
- Where a and b are given two strings.
- cost[i] represents the cost for alphabet i.
- Time Complexity = O(N*M)

Tushar and CAT preparations

Let's use binary search approach. For given number of books (say, x) let's find the minimal number of money needed to make them. Say, for one book Tushar needs cb blue papers, cs silver papers, cc cyan papers. So for x books he needs: $cb \cdot x$, $cs \cdot x$ and $cc \cdot x$ papers (by types). Since he already has nb, ns and ncpapers, so he needs to buy:

- Blue paper: $max(0, cb \cdot x - nb)$,
- Silver paper: $max(0, cs \cdot x - ns)$,
- Cyan paper: $max(0, cc \cdot x - nc)$.

So the formula to calculate money to make x books is:
$f(x) = max(0, cb \cdot x - nb) \cdot pb + max(0, cs \cdot x - ns) \cdot ps + max(0, cc \cdot x - nc) \cdot pc$

Obviously, the function $f(x)$ is monotonic (increasing). So it is possible to use binary search approach to find largest x such that $f(x)$ ler.

Aman, Bhavya and Chocolates

Consider a bipartite graph. In each part (we call them first and second part) there are $L = 2 \times 105$ vertices numbered from 1 to L. For each point (x, y) add an edge between vertex number x from the first part and vertex number y from the second part.
In this problem, we want to color edges with two colors so that the difference between the number of blue edges connected to a vertex and the number of red edges connected to it be at most 1.
Doing such thing is always possible.
We prove this and solve the problem at the same time with induction on the number of edges :
If all vertices have even degree, then for each component there is an Eulerian circuit, find it and color the edges alternatively_ with blue and red. Because graph is bipartite, then our circuit is an even walk and so, the difference between the number of blue and red edges connected to a vertex will be 0.
Otherwise, if a vertex like v has odd degree, consider a vertex like u that there is and edge between v and u. Delete this edge and solve the problem for the rest of the edges (with the induction definition) and then add this edge and if the number of red edges connected to u is more than the blue ones, then color this edge with blue, otherwise with red.
You can handle this add/delete edge requests and find odd vertices with a simple set.