ACM ICPC

UM Practice Contest 1

September 15

Problems

A	A+B Problem
B	Shortest Path
C	Chessboard Path Sums
D	Knapsack Problem
E	Rings

Problem A
A+B Problem

Given two non-negative integers, you are asked to calculate and print out $a + b$.

Input

Two integers a, b ($0 \leq a, b \leq 10^9$).

Output

Output $a + b$.

Sample Input

```
1 2
```

Sample Output

```
3
```

Hint

Q: Where are the input and the output?

A: Your program shall always read input from stdin (Standard Input) and write output to stdout (Standard Output). For example, you can use 'scanf' in C or 'cin' in C++ to read from stdin, and use 'printf' in C or 'cout' in C++ to write to stdout.

You shall not output any extra data to standard output other than that required by the problem; otherwise you will get a "Wrong Answer".

User programs are not allowed to open and read from/write to files. You will get a "Runtime Error" or a "Wrong Answer" if you try to do so.

Here is a sample solution using C++/ G++:

```
#include <iostream>
using namespace std;
int main(){
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
}
```

It's important that the return type of main() must be int when you use G++/ GCC, or you may get compile error.

Here is a sample solution using C/ GCC:

```
#include <stdio.h>
int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d\n", a + b);
    return 0;
}
```

Here is a sample solution using Java:

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String args[]) throws Exception {
        Scanner cin = new Scanner(System.in);
        int a = cin.nextInt(), b=cin.nextInt();
        System.out.println(a + b);
    }
}
```

Problem B
Shortest Path

This is a standard single source shortest path problem!

Given a directed graph with vertices, and a source vertex, find all shortest path distances from the given source vertex to all others.

Input

The first line contains three integers n, m, s $(2 \leq n \leq 200000, 0 \leq m \leq 500000, 1 \leq s \leq n)$, where n is the number of vertices, m is the number of edges, and s is the index of the source vertex.
The following m lines, each containing 3 integers a_i, b_i, c_i $(1 \leq a_i, b_i \leq n)$, represents a directed edge from a_i to b_i with length c_i.

For B-easy, all $c_i = 1$.
For B-hard, $1 \leq c_i \leq 10^8$.

Output

Output contains n lines, and the i-th line is an integer representing the shortest path length from s to i. If there is no such path, output -1 for that path instead.

Sample Input

```
3 1 1
1 2 10
```

Sample Output

```
0
10
-1
```

Problem C
Chessboard Path Sums

Alice and Bob are arguing over who will hold the title of the best problem-solver at UM. They already think it must be one of them, but which one? In a friendly competition, they decide to settle the score by solving each other's toughest problem.

Alice: Here is mine: you are given an $n \times m$ chessboard, where each position contains a positive integer. Your task is to walk your pawn through the board and to accumulate the maximum possible sum of all the values that you touch on your way. The only allowed moves you can make are one step down, or one step right. You must also start from the top left corner, and finish at the bottom right corner of the chessboard.

Bob: Hah! This is an easy dynamic programming problem. We can let $f[i][j]$ represent the maximum possible sum from the top left corner to position (i, j), then...

Alice: OK, I already know you got it. But what happens if you do the walk twice?

Bob: Then the answer is of course twice as much!

Alice: No, because, if you visit the same position twice, that value can be only be counted once.

Bob: Erm... OK, then this is still a dynamic programming problem, we let $f[i][j][k][l]$ represent the maximum possible sum of two paths, both start from the top left corner, and one ends at (i, j), and the other ends at (k, l).

Alice: OK, this is correct. But what if you can go t times?

Bob: We can still use dynamic programming. Let ...

Alice: Wait a second. How large is your array then?

Bob: It is $(n \times m)^t$. Well, I can bring it down to $(n \times m) \times (\min(n, m))^t$.

Alice: But it is still exponential to the input size.

Bob: Yes...

Bob now feels utterly lost and is asking for your help!

Input

The first line contains three integers, n, m, t $(1 \leq n, m \leq 100)$, where n by m is the size of the board, and t is the number of paths to take.
The following n lines, each contains m integers - the matrix. All the integers are positive and smaller than or equal to 1000.

For C-easy, $t = 1$.
For C-hard, $1 \leq t \leq 100$.

Output

A single integer: the maximum possible sum of t paths.

Sample Input

```
3 3 2
1 2 1
2 3 0
1 0 1
```

Sample Output

```
10
```

Explanation

```
1 2 1
2 3 0
1 0 1
```

```
1 2 1
2 3 0
1 0 1
```

Above are two paths from start to finish, which accumulate the largest possible sum. The red areas on path two are ones that were already counted for path #1 and therefore cannot be reused. The sum in the teal-colored areas is 10, which is the answer to sample input.

Problem D
Knapsack Problem

You are given N items, each item having a value of w_i and a volume $of\ v_i$. Each item can be used c_i times.

Your task is to determine which item, and how many of each item, to include in a collection, so that the total *volume* of collection is less than or equal to the given limit V, and that the total *value* of your collection is as large as possible.

Input

The first line contains two integers: $n\ (1 \leq n \leq 200)$ - the number of unique items, and $V(1 \leq V \leq 10^5)$ - the volume of the knapsack. The next n lines, are descriptions of each item, which are three integers: $v_i, w_i, c_i\ (1 \leq v_i, w_i \leq 1000)$, representing volume, value, and quantity of that item available.

For D-easy, $1 \leq c_i \leq 5$.
For D-hard, $1 \leq c_i \leq 2000$.

Output

A single integer represents the maximum value you can get.

Sample Input

```
3 100
10 20 5
100 100 1
80 80 1
```

Sample Output

```
120
```

Explanation of Sample

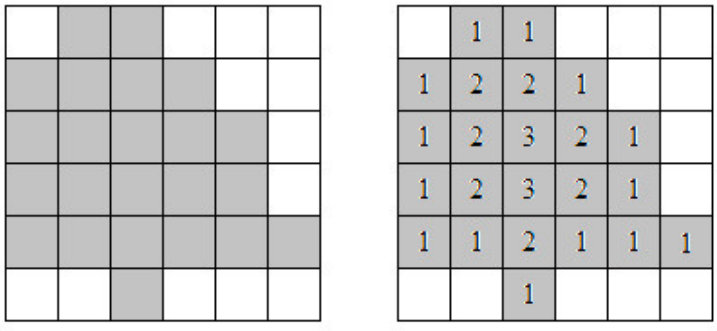
We can use the first item two times, and the third item one time. Resulting total *volume* of this collection is 100, which does not exceed the given volume of the knapsack, and total *value* is 2 * 20 + 80 = 120. For example, if we were to choose item two, it will have filled up the entire knapsack, with total value being only 100, lesser than 120. 120 is the better solution, and is also the best solution for this case – there is no other possible collection that exceeds 120.

Problem E
Rings

Dee Siduous is a botanist who specializes in trees. A lot of her research has to do with the formation of tree rings, and what they say about the growing conditions over the tree's lifetime. She has a certain theory and wants to run some simulations to see if it holds up to the evidence gathered in the field.

One thing that needs to be done is to determine the expected number of rings given the outline of a tree. Dee has decided to model a cross section of a tree on a two dimensional grid, with the interior of the tree represented by a closed polygon of grid squares. Given this set of squares, she assigns rings from the outer parts of the tree to the inner as follows: calling the non-tree grid squares "ring 0", each ring n is made up of all those grid squares that have at least one ring $(n - 1)$ square as a neighbor (where neighboring squares are those that share an edge).

An example of this is shown in the figure below.



Most of Dee's models have been drawn on graph paper, and she has come to you to write a program to do this automatically for her. This way she'll use less paper and save some … well, you know.

Input

The input will start with a line containing two positive integers n, m, specifying the number of rows and columns in the tree grid. After this will be n rows containing m characters each. These characters will be either 'T' indicating a tree grid square, or '.'.

For E-easy, $n = 1, 1 \leq m \leq 100$.
For E-hard, $1 \leq n, m \leq 100$.

Output

Output a grid with the ring numbers. If the number of rings is less than 10, use two characters for each grid square; otherwise use three characters for each grid square. Right justify all ring numbers in the grid squares, and use '.' to fill in the remaining characters.

If a row or column does not contain a ring number it should still be output, filled entirely with '.'s.

Sample Input

```
6 6
.TT...
TTTT..
TTTTT.
TTTTT.
TTTTTT
..T...
```

Sample Output

```
...1.1.....
.1.2.2.1....
.1.2.3.2.1..
.1.2.3.2.1..
.1.1.2.1.1.1
.....1......
```