Due: Thursday, August 9th, 6PM
Worth: 4%.
submissions: Use CDF submit command. Assignment name is PA2.
File name: PA2sol.c, PA2sol.cc, PA2sol.java or PA2sol.py
Any submission that:

- does not compile,
- produces a runtime error,
- does not adhere to the precise input/output format,

will not get any marks. Marks are earned based on passing (producing correct answer for) 20 test cases. For each test case that your program produces the correct output in the allocated runtime you get 5% of the assignment's mark.
You are free to write your answers in Python, Java, or C/C++ but be warned that Python programs might be at a disadvantage in terms of speed.
The runtime limit of the problem is there to test if you have implemented the fastest algorithm for the problem and is designed to be tight. A suboptimal algorithm, or a particularly poor implementation is not supposed to pass all the tests. (But if you can't figure out or implement the fastest algorithm feel free to submit a slower implementation; you will get the marks from the easier test cases.)

Question #1: The Grumkin and the Board

Grumkins are truly fascinating creatures. They are magical and obsessed with collecting big numbers. Their natural habitat is $(n \times m)$ boards and their movement is methodological if erratic.
Specifically, a grumkin starts its life on the upper left square of an $n \times m$ board. The grumkin then moves, slowing making its way to the lower right square of the board. There is a number between -1000 and 1000 at each square of the board and the grumkin "collects" the number in each of the squares it passes through. When it reaches the lower right corner the grumkin dies; the goal of the grumkin is to maximize the sum of the numbers it collects during its life. How a grumkin moves through the board is as follows. If the grumkin is currently at square (i, j) of the board (the squares are from the upper left $(1, 1)$ to the lower right (n, m)) it can magically jump to any of the squares:

- the square to the right if it exists, i.e. $(i, j + 1)$ if $j < m$;
- the square right below if it exists, i.e. $(i + 1, j)$ if $i < n$;
- the first square on the lower row if it exists, i.e. $(i + 1, 1)$ if $i < n$;
- any square to the right on the same row if its column number is a multiple of the current column number, i.e. (i, l) for any $l \leq m$ such that $l = kj$ for some integer $k > 1$.

Which of the above squares the grumkin jumps to is up to it. Your task is to write a program that gets n and the numbers on the board and outputs the maximum sum of numbers the grumkin can collect.

Input format: You algorithm should read its input from the standard input. The first line of input will have two numbers n, m, the size of the board. This will be followed by n lines each containing m numbers separated by space. The jth number on the ith of these lines will be the number on the square (i, j) of the board.

Conditions: The size of the board $n, m \leq 1000$ and the numbers on the board are between -1000 and 1000 inclusive.

Runtime: Your algorithm should take no more than 6 seconds on each input on the cdf servers. This is the sum of the "user time" and "system time" which you can see by using the "time" command on the cdf machine.

Output format: Your output should be written on the standard output and should have only one line with a single number, the sum of numbers the grumkin collects.

Sample input and output: Here are 4 sample inputs with correct output for each.

Sample Input	Correct output
1 5 2 10 -10 -3 10	19
3 3 1 1 -1 1 1 -1 -1 -1 2	5
2 5 5 0 5 1 0 1 2 3 4 5	26
2 2 -1 -1 -1 -1	-3
5 5 -144 -757 804 422 -988 -252 -732 5 778 377 -263 -291 -232 -246 743 794 129 621 -311 -475 -51 64 726 386 -97	5552

In the first input the best path the grumkin can take is $(1,1) \to (1,2) \to (1,4) \to (1,5)$. Notice how the grumkin ends up collecting some negative numbers.
In the second input the best path is $(1,1) \to (1,2) \to (2,1) \to (2,2) \to (3,2) \to (3,3)$.