University of Cape Town ~ Department of Computer Science
Computer Science 3003S Theory of Algorithms ~ 2014

Practical Test 1 Session 1 Question 2

50 Marks

You may consult the electronic Java and C/C++ API documentation (docs.cs.uct.ac.za), and submit to the automatic marker via Vula (vula.uct.ac.za), but nothing else! You may NOT use your class notes, textbooks, internet or files on your flash disk, hard drive, etc.

You may use only the computers in the lab. No tablets, headphones or other personal devices are permitted.

File names

- Use `scoring.c` if you are writing your program in C.
- Use `scoring.cpp` if you are writing your program in C++.
- Use `Scoring.java` if you are writing your program in Java.

Note that case matters.

Submission

The automatic marker contains a submission entry bearing the same title as this question sheet.

Submit your source file within a single compressed, '.ZIP', archive.

Make sure you create a '.ZIP' archive, not a gzipped, '.gz', or tar-gzipped, '.tgz', or other kind of file.

Make sure your source file is the only item within the archive. Especially, avoid submitting an archive containing a folder containing the file.

When submitting a Java source file copied from an editor like Eclipse or Netbeans, please remove any package line that may appear at the beginning of the code.

Scoring

Each test case that is answered correctly will earn 10 points.

Question continues on next page

Problem Description

Write a program that accepts as input a list of N positive integers, $V_1,\ldots,V_N$, and a target score, T. Starting with a score of 1 point, the program will process the list of numbers in order, in each case, choosing whether to add the number to the current score, or to multiply the current score by it. The object is to find the maximum score that can be achieved that is *less* than T.

Examples:

Given N = 4 numbers, <4, 2, 3, 5>, and a target of T = 40, one possible score is achieved as follows:

Starting with 1,

* 4 = 4
* 2 = 8
+ 3 = 11
* 5 = 55

But 55 is larger than the target of 40.

The maximum score (less than 40) that can be achieved is in fact 35. It is obtained as follows:

Starting with 1,

+ 4 = 5
* 2 = 10
* 3 = 30
+ 5 = 35

Input and Output

Program input and output will make use of `stdio` streams (`System.in` and `System.out` in Java) i.e. not file I/O.

Input consists of a series of integer values, each on a separate line. The first value is for N, the number of integers in the list, followed by the values for those integers ($V_1,\ldots,V_N$), followed by the value for the target score, T.

Constraints:

$1 \le N \le 20$
$1 \le V_i \le 1{,}000$
$1 \le T \le 1{,}000{,}000$

Output consists of a single integer (the maximum score which can be achieved that is *less* than T) followed by a line break --- in Java, for example, use `System.out.println`, not `System.out.print`. The automatic marker expects the output in this precise form.

Sample Input:

```
4
4
2
3
5
40
```

Sample output:

```
35
```

END

University of Cape Town ~ Department of Computer Science
Computer Science 3003S Theory of Algorithms ~ 2014

Practical Test 2 Session 1 Question 1

Time: 120 minutes

Problem Description

A collection of precious gems is being sold. In an attempt to prevent a single buyer from purchasing them all, the cost increases, greater than linearly, with the number of gems bought.

The total cost to buy K gems is f(K) where f(K) is defined as the sum of j * (K/j) for all integers 0 < j < K. *(Note that K/j here uses integer division, so the result has no fractional part. For example: 8/3 would equal 2.)*

As an example, to buy 5 gems, the cost, f(5), would be calculated as:

1 * (5/1) + 2 * (5/2) + 3 * (5/3) + 4 * (5/4) = 1*5 + 2*2 + 3*1 + 4*1 = 16

Write a program that, given an amount of currency, N, calculates the greatest value of K for which f(K) ≤ N i.e. Write a program that calculates the largest number of gems that can be bought for a given amount of money.

Constraints:

1 ≤ K ≤ 1,000,000
This means that the input value N is bounded by:
1 ≤ N ≤ f(1,000,000) i.e. 1 ≤ N ≤ 822467118437

Example:

Given N = 30:

f(6) = 1 * (6/1) + 2 * (6/2) + 3 * (6/3) + 4 * (6/4) + 5 * (6/5)
= 1*6 + 2*3 + 3*2 + 4*1 + 5*1
= 27

f(7) = 1 * (7/1) + 2 * (7/2) + 3 * (7/3) + 4 * (7/4) + 5 * (7/5) + 6 * (7/6)
= 1*7 + 2*3 + 3*2 + 4*1 + 5*1 + 6*1
= 34

So the maximum number of gems that could be purchased would be 6.

Note that the function f(K) is *strictly increasing*.

Note that values used in this question can be larger than the maximum value of a 32 bit integer type, requiring the use of 64 bit integer types (long in Java, long long in C and C++).

Question continues on next page

File names

- Use `pricing.c` if you are writing your program in C.
- Use `pricing.cpp` if you are writing your program in C++.
- Use `Pricing.java` if you are writing your program in Java.

Please remember to zip your file.

Input and Output

Program input and output will make use of `stdio` streams (`System.in` and `System.out` in Java) i.e. not file I/O.

Input consists of a single line containing a single integer value, N, the maximum amount of currency that can be spent.

Output consists of a single integer, K (the maximum number of gems that can be purchased), followed by a line break --- in Java, for example, use `System.out.println`, not `System.out.print`. The automatic marker expects output in this precise form.

Sample Input:

```
30
```

Sample output:

```
6
```

Scoring

Each test case that is answered correctly will earn 10 points.

END

University of Cape Town ~ Department of Computer Science
Computer Science 3003S Theory of Algorithms ~ 2014

Practical Test 2 Session 1 Question 2

Time: 120 minutes

Problem Description

A lumber yard has a stock of N long wooden planks with lengths L_1, ..., L_N. Planks can be divided into shorter planks --- a plank of length 12, for example, can be divided into three shorter planks, each of length 4 --- but separate pieces can't be joined to form longer planks.

A large order has been placed, requesting that at least K planks of equal length, M, be delivered.

Write a program that, given K and the lengths of planks in stock, L_1, ..., L_N, determines the maximum possible value for M.

Note that as the length increases, the number of planks that can be made will decrease.

Example

You are given N = 4 planks with lengths 10, 14, 15, 11. The order requests a minimum of 6 planks.

The order can be fulfilled with 6 planks of length 7 each:

- The plank of length 10 is divide into one plank of length 7, and one of length 3 (which is discarded).
- The plank of length 14 is divide into two planks of length 7.
- The plank of length 15 is divide into two planks of length 7, and one of length 1 (which is discarded).
- The plank of length 11 is divide into one plank of length 7, and one of length 4 (which is discarded).

The order can’t be fulfilled with planks of length 8 or more, so 7 is the maximum length.

Note that although the discarded pieces have a combined length greater than 7, they can't be combined to form a longer plank.

File names

- Use `dividing.c` if you are writing your program in C.
- Use `dividing.cpp` if you are writing your program in C++.
- Use `Dividing.java` if you are writing your program in Java.

Please remember to zip your file.

Question continues on next page

Input and Output

Program input and output will make use of `stdio` streams (`System.in` and `System.out` in Java) i.e. not file I/O.

Input consists of a series of integer values, each on a separate line. The first value is N, the number of planks in stock, followed by the lengths of those planks, L_1, ..., L_N, followed by K, the minimum number of planks required.

Output consists of a single integer, M (the maximum possible length that will allow K planks to be delivered), followed by a line break --- in Java, for example, use `System.out.println`, not `System.out.print`. The automatic marker expects output in this precise form.

Constraints:

$1 \le N \le 10{,}000$
$1 \le L_i \le 1{,}000{,}000{,}000$
$1 \le K \le 10{,}000{,}000$
The answer, M, will be bounded by:
$1 \le M \le 10{,}000{,}000$

Sample Input:

```
4
10
14
15
11
6
```

Sample output:

```
7
```

Scoring

Each test case that is answered correctly will earn 10 points.

END

Student Number:		Finish Time:	
Signature:			

University of Cape Town ~ Department of Computer Science
Computer Science 3003S Theory of Algorithms ~ 2014

Procedure

You may consult the electronic Java and C/C++ API documentation (docs.cs.uct.ac.za), and submit to the automatic marker via Vula (vula.uct.ac.za), but nothing else! You may NOT use your class notes, textbooks, internet or files on your flash disk, hard drive, etc.

You may use only the computers in the lab. No tablets, headphones or other personal devices are permitted.

Please sign and return this sheet when you have finished.

Submission

The automatic marker contains a submission entry for each question. Entry and question bear the same name.

Submit your source file within a single compressed, '.ZIP', archive.

Make sure you create a '.ZIP' archive, not a gzipped, '.gz', or tar-gzipped, '.tgz', or other kind of file.

Make sure your source file is the only item within the archive. Especially, avoid submitting an archive containing a folder containing the file.

When submitting a Java source file copied from an editor like Eclipse or Netbeans, please remove any package line that may appear at the beginning of the code.

Practical Test 3 Session 1 Question 1

100 Marks

Each test case that is answered correctly will earn 10 marks.

File names

- Use cycling.c if you are writing your program in C.
- Use cycling.cpp if you are writing your program in C++.
- Use Cycling.java if you are writing your program in Java.

Note that case matters.

Problem Description

The scenario:

- N signs have been set up along a straight road, each displaying a number (which can be positive or negative).
- A cyclist must choose a position on the road at which to start cycling and a position at which to stop.
- The cyclist starts with 0 points. As they cycle from the chosen start position to the end position, they must add the number on any sign that they pass (whether positive or negative) to their score.
- The challenge is to choose a start and end position such that the point score is maximised.

Question continues on next page

Write a program that, given a series of N road sign numbers in the order in which they appear, calculates the maximum point score that can be achieved.
Example:

Assume a road with N = 6 consecutive signs with the following values:

Sign	S_1	S_2	S_3	S_4	S_5	S_6
Value	1	-2	4	-1	5	-3

Choosing to start cycling before the first sign and to stop after the fifth sign, would give a score of 1 + (-2) + 4 + (-1) + 5 = 7 points.

This is not, however, the maximum achievable score. Starting before the third sign and stopping after the fifth would render 4 + (-1) + 5 = 8 points. Thus the correct answer for this case would be 8.

Note that it is possible to start AND stop before the first sign, (or any other sign), giving a score of 0. While not true for this example, there may be situations in which that is the best choice.

Input and Output

Input consists of a series of integer values, each on a separate line. The first value is N, the number of signs on the road, followed by the point values P_1, …, P_N, for those signs. The values of the signs are given in the order they appear on the road.

Constraints:

$1 \le N \le 2{,}000$

$-1{,}000{,}000 \le P_i \le 1{,}000{,}000$ (for $1 \le i \le N$)

Output consists of a single integer, K (the maximum point score that can be achieved), followed by a line break --- in Java, for example, use `System.out.println`, not `System.out.print`. The automatic marker expects output in this precise form.

The maximum achievable point score will fit within a 32 bit signed integer type.

Sample Input:

```
6
1
-2
4
-1
5
-3
```

Sample output:

```
8
```

Question continues on next page

Practical Test 3 Session 1 Question 2

50 Marks

Each test case that is answered correctly will earn 5 marks.

File names

- Use path **.** `c` if you are writing your program in C.
- Use path **.** `cpp` if you are writing your program in C++.
- Use Path.`java` if you are writing your program in Java.

Note that case matters.

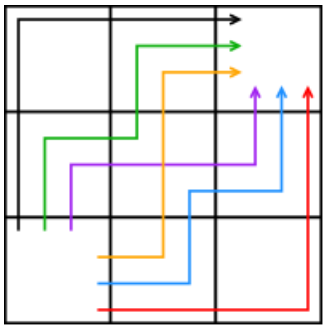
Problem Description

Write a program that computes the number of paths that a robot may take when navigating through a given terrain.

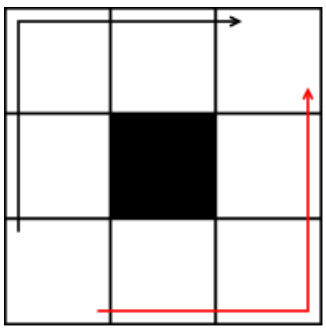
- A terrain is represented by an NxN grid of squares.
- The robot starts in the lower left square of grid, referred to as square (1, 1), and needs to move to the upper right square, referred to as (N, N).
- A terrain also contains K obstacles, each of which blocks a single square on the grid.
- The robot cannot move into a square containing an obstacle, so these reduce the number of possible paths through the grid.
- No two obstacles block the same square, and the starting and ending squares will never contain obstacles.
- The robot can only move a single square up or right with each step.

Example:

Given a 3x3 grid with no obstacles, there are a total of 6 paths from the bottom left square to the top right, shown below:

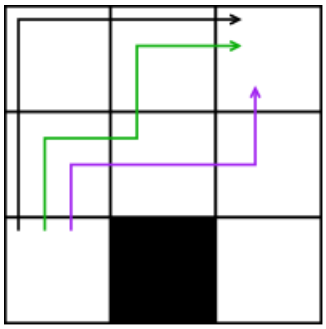


Given the same grid with one obstacle on square (2, 2) there are 2 paths:



Question continues on next page

Given the same grid with one obstacle on square (2, 1) there are 3 paths:



Input and Output

Input consists of a series of lines, each containing up to two integer values.

- The first line of input contains a single integer, N, representing the size of the grid.
- The second line contains a single integer K, representing the number of obstacles on the grid.
- The following K lines of input each contain two integers separated by a space. Each of these lines represents the coordinates on the grid of one of the obstacles.

Output consists of a single integer P (the number of paths from the bottom left grid square, to the top right grid square), followed by a line break --- in Java for example, use `System.out.println`, not `System.out.print`. The automatic marker expects output in this precise form.

Constraints:

$1 \leq N \leq 20$
$0 \leq K \leq N$
All obstacles will have coordinates within the dimensions of the grid.
The answer, P, will be bounded by:
$0 \leq P \leq 1{,}000{,}000{,}000$

Sample Input:

```
3
1
2 1
```

Sample output:

```
3
```

END

Question continues on next page