The 4th Annual Newport News Computer Challenge

Wednesday, February 16, 2005

Team Packet

Java Problems

The 4th Annual Newport News Computer Challenge

Wednesday, February 16, 2005

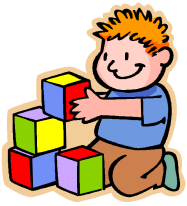
Java Problems

```
1. INPUT n
2. PRINT n
3. IF n = 1 THEN STOP
● ● ●
```

The 3n + 1 Problem ~ 10 points

Freddie's Chair ~ 20 points

ISBN Numbers ~ 20 points



The Blocks Problem ~ 30 points

The 4th Annual Newport News Computer Challenge

Wednesday, February 16, 2005

The 3n + 1 Problem (10 points)

Consider the following algorithm, expressed in psuedo-code of no particular computer language:

1. INPUT n
2. PRINT n
3. IF n = 1 THEN STOP
4. IF n is odd THEN n ← 3n + 1
5. ELSE n ← n/2
6. GOTO 2

(The ← symbol means assignment.)

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1.

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that $0 < n <$ 1,000,000 (and, in fact, for many more numbers than this.)

Given an input n, it is possible to determine the number of numbers printed (including the 1). For a given n this is called the *cycle-length* of n. In the example above, the cycle length of 22 is 16.

For any two numbers i and j you are to determine the maximum cycle length over all numbers between i and j inclusive.

Sample run:

```
The 3n + 1 Problem, by [your school & team name here]
Enter two positive integers, on separate lines below, zeros to quit.
1
10
Maximum cycle length: 20

Enter two positive integers, on separate lines below, zeros to quit.
100
200
Maximum cycle length: 125

Enter two positive integers, on separate lines below, zeros to quit.
201
210
Maximum cycle length: 89

Enter two positive integers, on separate lines below, zeros to quit.
900
1000
Maximum cycle length: 174

Enter two positive integers, on separate lines below, zeros to quit.
0
0

Press Enter to continue . . .
```

Newport News Public Schools

The 4th Annual Newport News Computer Challenge

Wednesday, February 16, 2005

Freddie's Chair (20 points)

On certain days in Freddie's math class when too many students were absent due to SOL testing, Mrs. Magillicuddy would have the students who were left in class play a game she called "Chair Elimination".

The game of "Chair Elimination" works like this: The center of the room is cleared and students arrange their chairs in a circle in the center of the room, one for each student participating in the game, all chairs facing inward with nobody sitting in the chairs. Chairs are numbered in a clockwise direction starting with one and ending with the number of chairs.

The game leader (in this case, Mrs. Magillicuddy) randomly picks two numbers. The first number, called the "starter number", is a number of one of the chairs. The second number, called the "elimination number", is a number between 3 and 100 inclusive.

The game leader announces the two numbers and students are then told, "Ladies and gentlemen, have a seat," at which point everyone scrambles for a chair. The person who ends up in the chair whose number is the "starter number" is called the "starter".

Beginning with the "starter", students count aloud and clockwise around the circle. The "starter" says "one", the person to the immediate left of the "starter" says "two", the person to the immediate left of the person who said "two" says "three", and so forth. When some unfortunate student says the "elimination number", that student must get up, pick up his or her chair, and leave the circle. Chairs are scooted inward slightly to tighten up the circle again. The person who was on the immediate left of the person who got eliminated becomes the new "starter" and the counting process begins with "one" again. When all persons except one have been eliminated, the last remaining person becomes the winner.

Mrs. Magillicuddy always had plenty of great prizes for winners in this game such as pencils, pens, candy bars, and sometimes even a shiny new Susan B. Anthony silver dollar. But one day everyone realized that Freddie was winning almost all the time so Mrs. Magillicuddy said they couldn't play "Chair Elimination" anymore and that they had to do worksheets instead. What nobody realized was that Freddie had a programmable earring into which he could enter the number of chairs, the "starter number", and the "elimination number" and then calculate the number of the chair that would win. He would then quickly make his best effort to sit in that chair before anyone else.

Your job is to write the program that Freddie used to win. Please reject invalid input and only output the winning chair.

> You may receive up to 15 points for a program that runs correctly as specified and an additional 5 points (for a total of 20 points) if your program uses a GUI.

Samples:

10 chairs, starter number 4, elimination number 5
The eliminated chairs in order of elimination are 8 3 9 5 2 1 4 7 10 and the winning chair is 6.

15 chairs, starter number 9, elimination number 17
The eliminated chairs in order of elimination are 10 13 2 7 15 9 6 8 14 11 1 3 5 12 and the winning chair is 4.

Newport News Public Schools

The 4th Annual Newport News Computer Challenge

Wednesday, February 16, 2005

ISBN Numbers (20 points)

An ISBN (International Standard Book Number) has ten digits. The first nine digits may have values from '0' to '9'; they identify the country in which the book was printed, the publisher, and the individual book. The tenth digit is a "check digit" assigned in such a way that the number $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}$ has the property:

$$(10d_1 + 9d_2 + 8d_3 + 7d_4 + 6d_5 + 5d_6 + 4d_7 + 3d_8 + 2d_9 + d_{10}) \bmod 11 = 0$$

"mod" stands for modulo division (same as % in C++ or Java). If d_{10} needs the value 10 to balance the check digit equation, then the character 'X' is used. For example, 096548534X is a valid ISBN.

Write a Java application that takes as input a 9-digit string of characters.

If the input string is not 9 numerical digits (each being 0 through 9) then a message is returned stating that the input is invalid.

Otherwise your problem will display the 10th ISBN character.

Samples:

input	output
12345678	Invalid ISBN Entry.
135792468	2
192837465	4
192837462	X

You may receive up to 15 points for a program that runs correctly as specified above.
You may receive up to 5 additional points (for a total of up to 20 points) if your program uses a GUI.

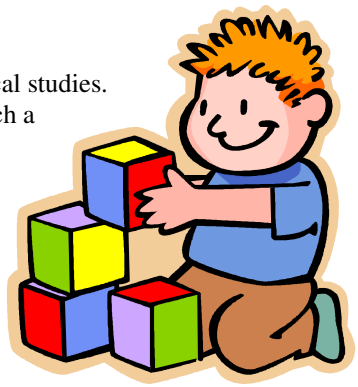
Newport News Public Schools

The 4th Annual Newport News Computer Challenge

Wednesday, February 16, 2005

The Blocks Problem (30 points)

Background



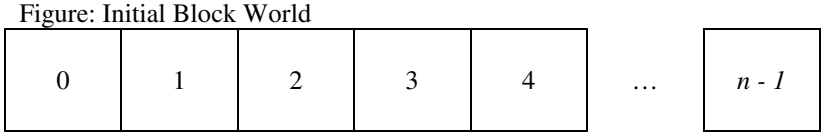
Many areas of Computer Science use simple, abstract domains for both analytical and empirical studies. For example, an early AI study of planning and robotics (STRIPS) used a block world in which a robot arm performed tasks involving the manipulation of blocks.

In this problem you will model a simple block world under certain rules and constraints. Rather than determine how to achieve a specified state, you will "program" a robotic arm to respond to a limited set of commands.

The Problem

The problem is to parse a series of commands that instruct a robot arm in how to manipulate blocks that lie on a flat table. Initially there are *n* blocks on the table (numbered from 0 to *n* - 1) with block b_i adjacent to block b_{i+1} for all $0 \le i \le n-1$ as shown in the diagram below:

Figure: Initial Block World



The valid commands for the robot arm that manipulates blocks are:

* move *a* onto *b*

where *a* and *b* are block numbers, puts block *a* onto block *b* after returning any blocks that are stacked on top of blocks *a* and *b* to their initial positions.

* move *a* over *b*

where *a* and *b* are block numbers, puts block *a* onto the top of the stack containing block *b*, after returning any blocks that are stacked on top of block *a* to their initial positions.

* pile *a* onto *b*

where *a* and *b* are block numbers, moves the pile of blocks consisting of block *a*, and any blocks that are stacked above block *a*, onto block *b*. All blocks on top of block *b* are moved to their initial positions prior to the pile taking place. The blocks stacked above block *a* retain their order when moved.

* pile *a* over *b*

where *a* and *b* are block numbers, puts the pile of blocks consisting of block *a*, and any blocks that are stacked above block *a*, onto the top of the stack containing block *b*. The blocks stacked above block *a* retain their original order when moved.

* quit

terminates manipulations in the block world.

Any command in which *a* = *b* or in which *a* and *b* are in the same stack of blocks is an illegal command. All illegal commands should be ignored and should have no affect on the configuration of blocks.

The Input

The input begins with an integer *n* on a line by itself representing the number of blocks in the block world. You may assume that $0 < n < 25$.

The number of blocks is followed by a sequence of block commands, one command per line. Your program should process all commands until the quit command is encountered.

You may assume that all commands will be of the form specified above. There will be no syntactically incorrect commands.

The Output

The output should consist of the final state of the block world. Each original block position numbered *i* ($0 \le i \le n-1$ where *n* is the number of blocks) should appear followed immediately by a colon. If there is at least a block on it, the colon must be followed by one space, followed by a list of blocks that appear stacked in that position with each block number separated from other block numbers by a space. Don't put any trailing spaces on a line.

There should be one line of output for each block position (i.e., *n* lines of output where *n* is the integer on the first line of input).

Sample Input

```
10
move 9 onto 1
move 8 over 1
move 7 over 1
move 6 over 1
pile 8 over 6
pile 8 over 5
move 2 over 1
move 4 over 9
quit
```

Sample Output

```
0: 0
1: 1 9 2 4
2:
3: 3
4:
5: 5 8 7 6
6:
7:
8:
9:
```

Using a GUI is not recommended for this problem. You probably don't have time. But if you do, sorry, there will be no extra points for it.