The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

Team Packet

C++
Problems

The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

C++ Problems

Distance 3D ~ 10 points
Divisors ~ 20 points
Smith Numbers ~ 20 points
Mouse Clicks ~ 30 points

The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

Distance 3D (10 points)

Write a C++ program that calculates the distance between a pair of points on a three dimensional Cartesian coordinate system.

The distance d between two points, $(x1,y1,z1)$ and $(x2,y2,z2)$, on a three dimensional Cartesian coordinate system is

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - x2)^2}$$

Input Specification

On one line enter three integers—the coordinates of the first point. On another line enter three integers—the coorindates of the second point.

Output Specification

Display a line that says "The distance from (x,y,z) to (a,b,c) is d.ddd" where x, y, z, a, b, and c stand for the actual integers input by the user and d.ddd stands for the distance between those two points with three decimal places displayed (0's if needed) and rounded to the nearest thousandth.

Sample runs (without the heading, etc.)

```
Enter coordinates of the first  point: 0 0 0
Enter coordinates of the second point: 0 0 0
The distance from (0,0,0) to (0,0,0) is 0.000

Enter coordinates of the first  point: 1 -2 3
Enter coordinates of the second point: 1 -2 4
The distance from (1,-2,3) to (1,-2,4) is 1.000


Enter coordinates of the first  point: 1 2 3
Enter coordinates of the second point: 4 5 6
The distance from (1,2,3) to (4,5,6) is 5.196


Enter coordinates of the first  point: 18 22 72
Enter coordinates of the second point: -122 -144 99
The distance from (18,22,72) to (-122,-144,99) is 218.826
```

The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

Divisors (20 points)

Mathematicians love all sorts of odd properties of numbers. For instance, they consider 945 to be an interesting number, since it is the first odd number for which the sum of its divisors is larger than the number itself.

To help them search for interesting numbers, you are to write a program that scans a range of numbers and determines the number that has the largest number of divisors in the range. Unfortunately, the size of the numbers, and the size of the range is such that a too simple-minded approach may take too much time to run. So make sure that your algorithm is clever enough to cope with the largest possible range in just a few seconds.

Input Specification

The first line of input specifies the number *N* of ranges, and each of the *N* following lines contains a range, consisting of a lower bound *L* and an upper bound *U*, where *L* and *U* are included in the range. *L* and *U* are chosen such that 1 <= L <= U <= 1,000,000,000 and 0 <= U - L <= 10,000.

Output Specification

For each range, find the number *P* which has the largest number of divisors (if several numbers tie for first place, select the lowest), and the number of positive divisors *D* of *P* (where *P* is included as a divisor). Print the text `'Between L and H, P has a maximum of D divisors.'`, where *L*, *H*, *P*, and *D* are the numbers as defined above.

Example input

```
3
1 10
1000 1000
999999900 1000000000
```

Example output

```
Between 1 and 10, 6 has a maximum of 4 divisors.
Between 1000 and 1000, 1000 has a maximum of 16 divisors.
Between 999999900 and 1000000000, 999999924 has a maximum of 192 divisors.
```

The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

Smith Numbers (20 points)

Background

While skimming his phone directory in 1982, Albert Wilansky, a mathematician of Lehigh University , noticed that the telephone number of his brother-in-law H. Smith had the following peculiar property: The sum of the digits of that number was equal to the sum of the digits of the prime factors of that number. Got it? Smith's telephone number was 493-7775. This number can be written as the product of its prime factors in the following way:

4937775 = 3 * 5 * 5 * 65837

The sum of all digits of the telephone number is 4+9+3+7+7+7+5=42, and the sum of the digits of its prime factors is equally 3+5+5+6+5+8+3+7=42. Wilansky was so amazed by his discovery that he named this type of number after his brother-in-law: Smith Numbers.

As this observation is also true for every prime number, Wilansky decided later that a (simple and unsophisticated) prime number is not worth being a Smith Number and he excluded them from the definition.

Problem

Wilansky published an article about Smith Numbers in the *Two Year College Mathematics Journal* and was able to present a whole collection of different Smith Numbers: For example, 9985 is a Smith Number and so is 6036. However, Wilansky was not able to give a Smith Number which was larger than the telephone number of his brother-in-law. It is your task to find Smith Numbers, including those which may be larger than 4937775, but you must do it like this:

Input

Prompt the user repeatly for a positive integer (0 to 2 billion). Input ceases when the user enters 0 or less.

Output

For every input value *n*, you are to compute the smallest Smith Number which is larger than *n* and display it, its prime factors, the sum of its digits, and the sum of the digits of its prime factors. But you must do so using the format of the Sample Run that follows.

Sample Run

```
Smith Number Tester. By C. Monroe

This program will find the next larger Smith Number
of any positive integer that you enter, and explain
why it is a Smith Number.

Enter your positive integer (0 or less quits): 1000
The next Smith Number larger than 1000 is 1086 because
1086 = 2 * 3 * 181
and
1+0+8+6 = 2+3+1+8+1 = 15
Enter your positive integer (0 or less quits): 4937774
The next Smith Number larger than 4937774 is 4937775 because
4937775 = 3 * 5 * 5 * 65837
and
4+9+3+7+7+7+5 = 3+5+5+6+5+8+3+7 = 42
Enter your positive integer (0 or less quits): 87654321
The next Smith Number larger than 87654321 is 87654327 because
87654327 = 3 * 29 * 131 * 7691
and
8+7+6+5+4+3+2+7 = 3+2+9+1+3+1+7+6+9+1 = 42
Enter your positive integer (0 or less quits): 0
Press any key to continue . . .
```

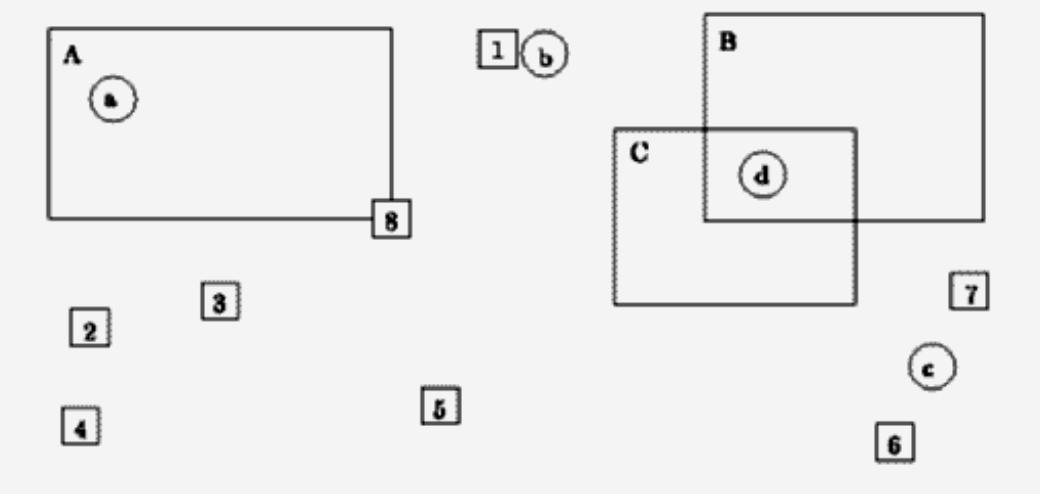
The 3rd Annual Newport News Computer Challenge

Thursday, February 19, 2004

Mouse Clicks (30 points)

A typical windowing system on a computer will provide a number of icons on the screen as well as some defined regions. When the mouse button is clicked, the system has to determine where the cursor is and what is being selected. For this problem we assume that a mouse click in (or on the border of) a region selects that region, otherwise it selects the closest visible icon (or icons in the case of a tie).

Consider the following screen. Rectangles represent regions. Squares represent icons. Circles represent mouse click locations.



A mouse click at ‘a’ will select region A. A mouse click at ‘b’ will select icon 1. A mouse click at ‘c’ will select icons 6 and 7. A mouse click at ‘d’ is ambiguous. The ambiguity is resolved by assuming that one region is in front of another. In the data files, later regions can be assumed to be in front of earlier regions. Since regions are labelled in order of appearance (see later) ‘d’ will select C. Note that regions always overlap icons so that obscured icons need not be considered and that the origin (0,0) is at the top left corner.

Write a program that will read in a series of region and icon definitions followed by a series of mouse clicks and return the selected items. Coordinates will be given as pairs of integers in the range 0..499 and you can assume that all icons and regions lie wholly within the screen. Your program must number all icons (even invisible ones) in the order of arrival starting from 1 and label regions alphabetically in the order of arrival starting from `A'.

Input

Input will consist of a series of lines. Each line will identify the type of data: I for icon, R for region and M for mouse click. There will be no separation between the specification part and the event part, however no icon or region specifications will follow the first mouse click. An I will be followed by the coordinates

of the centre of the icon, R will be followed by the coordinates of the top left and bottom right corners respectively and M will be followed by the coordinates of the cursor at the time of the click. There will always be at least one visible icon and never more than 25 regions and 50 icons. The entire file will be terminated by a line consisting of a single `#`.

Output

Output will consist of one line for each mouse click, containing the selection(s) for that click. Regions will be identified by their single character identifier, icon numbers will be written out right justified in a field of width 3, and where there is more than one icon number they will appear in increasing numerical order.

Sample input

```
I       216     28
R       22      19      170     102
I       40      150
I       96      138
I       36      193
R       305     13      425     103
I       191     184
I       387     200
R       266     63      370     140
I       419     134
I       170     102
M       50      50
M       236     30
M       403     167
M       330     83
#
```

Sample output

```
A
  1
  6   7
C
```