The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

Team Packet

C++
Problems

The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

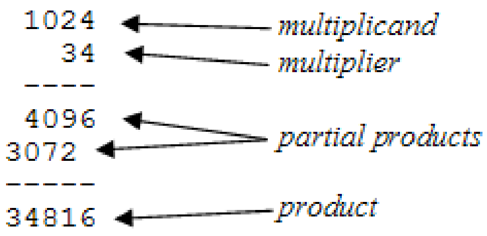
C++ Problems

Ugly Numbers ~ 10 points

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

The 3n + 1 Problem ~ 10 points

Credit Card Validation ~ 20 points



Long Multiplication ~ 20 points

The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

Ugly Numbers (10 points)

Ugly numbers are numbers whose only prime factors are 2, 3 or 5.

The sequence

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...

shows the first 11 ugly numbers. By convention, 1 is included.

Write a program that prompts the user to input a positive integer *n* and outputs the *nth* ugly number followed by the next four ugly numbers.

The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

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 <u>*i* and *j* inclusive.</u>

Sample run:

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

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

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

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

Enter two positive integers, zeros to quit.
0 0

Press Enter to continue . . .
```

The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

Credit Card Validation (20 points)

This problem lets you in on a little secret used by banks and lending institutions (called *the Luhn algorithm*, named after the IBM scientist Hans Peter Luhn who developed it in 1954) to determine whether or not a credit card number is valid. What we're talking about here is not whether a credit card number is currently being used by someone, but rather whether or not the number COULD be used by someone, should a bank or lending institution issue that number.

Below is the Luhn algorithm that tests for credit card validity:

Step	**Sample (using 5318-2795-1234-5678)**
1. Reverse the order of the digits of the card number.	8765 4321 5972 8135
2. Beginning with the second digit and continuing with every other digit…	These digits are 7, 5, 3, 1, 9, 2, 1, and 5.
3. Perform this function on each of these digits: double the digit and (if the doubled result has two digits) add the digits of the doubled result.	$7 \times 2 = 14, 1 + 4 = 5$ $5 \times 2 = 10, 1 + 0 = 1$ $3 \times 2 = 6$ $1 \times 2 = 2$ $9 \times 2 = 18, 1 + 8 = 9$ $2 \times 2 = 4$ $1 \times 2 = 2$ $5 \times 2 = 10, 1 + 0 = 1$
4. Find the sum of these results.	The sum of 1, 2, 4, 9, 2, 6, 1, and 5 is 30.
5. Find the sum of the other digits.	The sum of 3, 8, 7, 5, 2, 4, 6, and 8 is 43.
6. Add the two sums.	30 plus 43 is 73.
7. If the final result is divisible by 10, the credit card number is valid. If not, it is invalid.	73 is not divisible by 10. Therefore the credit card number is invalid.

Your job, however, is not to test for validity, but rather to allow the user to enter the first ***n-1*** digits of a ***n***-digit credit card number and then output the digit that, if used as the ***n***th digit, would make the card number valid. Input should allow dashes or spaces to be inserted anywhere in the number.

Samples:

First n-1 digits	nth digit
`5318-2795-1234-567`	`5`
`1234 12 123`	`2`
`000-000-100`	`8`
`000-000-1000`	`9`

The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

Long Multiplication (20 points)

Given two positive integers in either order, display the problem as long multiplication, each line correctly right-justified. The smaller integer should always be displayed as the multiplier. For example, to display 34 times 1024:

```
 1024  ← multiplicand
   34  ← multiplier
 ----
 4096  ← partial products
3072   ←
-----
34816  ← product
```

To get maximum points: Partial products equaling zero should be omitted. If the multiplier is a single digit, no partial products should be displayed. The length of the first dashed line (if there) should equal the number of digits in the multiplicand or the first partial product, whichever is larger. The length of the dashed line over the product should equal the number of digits in the product. Input should continue until at least one zero is entered. Negative numbers should be rejected. Do not worry about other invalid input. Do not worry about overflow.

```
Long Multiplication.  By the C++ team at Sample High School.
Please enter two positive integers to multiply, either one zero to quit.
-1 234
Please enter two positive integers to multiply, either one zero to quit.
1234 23456

     23456
      1234
     -----
     93824
    70368
   46912
  23456
  --------
  28944704

Please enter two positive integers to multiply, either one zero to quit.
20304 102

    20304
      102
    -----
    40608
  20304
  -------
  2071008

Please enter two positive integers to multiply, either one zero to quit.
1234 3

  1234
     3
  ----
  3702

Please enter two positive integers to multiply, either one zero to quit.
0 1

Press any key to continue...
```

The 9th Annual Newport News Computer Challenge

Thursday, February 18, 2010

C++ Ruberics for Teams

Ugly Numbers (10 points)

Heading displays program name and team's name.	1
Prompts user to input a positive integer *n*.	1
Accepts the positive integer without crashing.	1
Outputs the *nth* ugly number…	3
…followed by the next four ugly numbers. *Score here is 1 point per correct number displayed <u>minus</u> 1 point per incorrect number displayed. Displaying more than four numbers, correct or not, is a deduction of 1 point. Max 4, minimum 0.*	4
TOTAL	10

The 3n + 1 Problem (10 points)

One or more heading lines with problem title, school name, and team name.	1
User is prompted to type two positive integers or two zeros to quit.	1
Program accepts two positive integers or zeros without crashing. (on one or two lines okay)	1
Correct cycle length is displayed. • all four samples below, 5 points • at least two samples, 3 points	5
Cycle length is labeled.	1
If two zeros are entered, program terminates, otherwise repeats.	1
TOTAL	10

Credit Card Validation (20 points)

Heading displays program name.	1
Friendly prompt to enter all but the last digit of a credit card number.	1
Accepts the input without crashing and without ending immediately.	2
Rejects or ignores invalid characters (only digits, spaces, and dashes are allowed).	3
For invalid input, displays a message stating that the input is invalid.	3
For valid input, correctly displays the last digit.	10
TOTAL	20

Long Multiplication (20 points)

Heading displays program name and team name.	1
Prompts for two positive integers, either one zero to quit.	1
Re-prompts if either integer is negative.	1
Quits if either integer is zero (provided it doesn't quit for other values).	1
Redisplays multiplicand and multiplier correctly formatted, right-justified, larger on top, with line of dashes drawn underneath. (Correctly right-justifying all three lines sets the right margin by which all other subsequent alignment is determined. If this margin is not correct, all alignment/indention points below are lost.)	1
Correctly displays all partial products on each line below.	5
Correctly displays all partial products on each line below, correctly indented from the right margin set previously.	4
Correctly draws a line of dashes and displays the product underneath the partial products.	2
Correctly draws a line of dashes and displays the product underneath the partial products, right-justified using the right margin set previously.	2
Partial products equaling zero are omitted.	1
If the multiplier is a single digit, no partial product is displayed and only one line of dashes is drawn.	1
TOTAL	20