The Heritage Computer Challenge
2006
Heritage High School
Newport News, Virginia
Java Division

Welcome

Welcome to the Heritage Computer Challenge for 2006! You are to be commended for taking the time and making the effort to be here today. Have a great time and may all your programming efforts be successful!

~Mr. Charles F. Monroe, Contest Director

Instructions

The problems for this contest appear on the following pages, listed in order of difficulty. The maximum number of points you can earn is indicated under the title to each problem.

Problems are designed in the format used by The Great Computer Challenge, held annually each Spring at Old Dominion University. Some of these problems were actually used at the Great Computer Challenge in previous years.

How to save your work:
1. Create a folder on your personal drive K named hcc2006.
2. Create a workspace named hcc2006 and save it inside this folder.
3. Create all projects within this workspace.
4. Each solution should be saved as a project using the project name provided.

List of problems

PC	10 points
Smith Numbers	20 points
Palindromes	20 points
The Cat In The Hat	30 points

PC
(10 points)

Save in folder named: **PC**

Write a program that will print the word "PC" in large print (displayed in the console window, or in a JTextArea component if you choose to use a GUI although we don't advise it). The input to the program consists of two numbers (each between 5 and 20 inclusive) that tells you how many lines the letters are high. Both letters must be on the same bottom line. For example, if the input to the program were 5 and 8, the output would look like this:

```
               CCCCCCCC
             CCC      CCC
             CCC      CCC
PPPPPPPP     CCC
PPP   PPP    CCC
PPPPPPPP     CCC      CCC
PPP          CCC      CCC
PPP            CCCCCCCC
```

The Heritage Computer Challenge *2006*
Java Division

Smith Numbers
(20 points)

Save in folder named: **SmithNumbers**

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:

continued on the next page

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 Heritage Computer Challenge *2006*
Java Division

Palindromes
(20 points)

Save in folder named: **Palindromes**

A palindrome is a word, phrase, verse, paragraph, etc., which reads the same forwards or backwards (excluding punctuation, spacing, and capitalization). For example, the following are palindromes:

Madam, I'm Adam.
Poor Dan is in a droop.
Now Sir, a war is never even. Sir, a war is won.
Sue Zues.
Evade Dave.

Write a program which takes an arbitrary list of alphabetic characters and determines if it is a palindrome or not.

Input for each palindrome will not exceed 80 characters. Note that blanks (spaces), capitalization, and punctuation do not affect the determination of a palindrome. Numeric characters will not occur in the input. Your program should consider only one palindrome at a time, i.e., for each run of the program, there is only one palindrome to read.

Output should be PALINDROME if the expression is a palindrome, or NOT A PALINDROME if it is not.

Example:

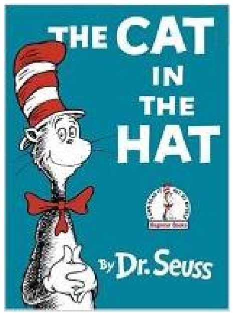
```
Palindrome Test
Not now, no strap parts on Won Ton.
PALINDROME
```

The Heritage Computer Challenge *2006*
Java Division

The Cat in the Hat
(30 points)

Save in folder named: **CatInHat**

The Cat in the Hat is a nasty creature,
But the striped hat he is wearing has a rather nifty feature.
With one flick of his wrist he pops his top off.
Do you know what's inside that Cat's hat?
A bunch of small cats, each with its own striped hat.
Each little cat does the same as line three,
All except the littlest ones, who just say ``Why me?"
Because the littlest cats have to clean all the grime,
And they're tired of doing it time after time!



A clever cat walks into a messy room which it needs to clean. Instead of doing the work alone, it decides to have its helper cats do the work. It keeps its (smaller) helper cats inside its hat. Each helper cat also has helper cats in its own hat, and so on. Eventually, the cats reach a smallest size. These smallest cats have no additional cats in their hats. These unfortunate smallest cats have to do the cleaning.

The number of cats inside each (non-smallest) cat's hat is a constant, *N*. The height of these cats-in-a-hat is $\frac{1}{N+1}$ times the height of the cat whose hat they are in. The smallest cats are of height one; these are the cats that get the work done. All heights are positive integers.

Given the height of the initial cat and the number of worker cats (of height one), find the number of cats that are not doing any work (cats of height greater than one) and also determine the sum of all the cats' heights (the height of a stack of all cats standing one on top of another).

Of course, only certain combinations of initial heights and worker cat counts are valid—but you can figure that out. Here are some sample combinations and results to help you out.

Height of the Initial Cat	Number of Worker Cats	Number of Cats that are Not Working	Height of the Stack of Cats
27	8	7	65
216	125	31	671
5764801	1679616	335923	30275911