The Heritage Computer Challenge
2005
Heritage High School
Newport News, Virginia

Java 

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 hcc2005.
2. Create a workspace named hcc2005 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.

********** Problems **********

Where is My Robot?

(10 points)

Project Name: **Robot**

The problem is one of finding the coordinates of the missing Robot. The Robot is known to have strted at the Cartesian coordinates of (0,0) and to have made a series of moves as dictated by his input sequence. Each step in the input sequence is one of the words north, south, east, and west. For an input of east, the Robot moves on unit in the positive x direction and of course no go west the Robot moves on unit in the negative x direction. Similarly a movement to the north casues the y value to increase by one while south causes the y value to decrease by one.

Given an input sequence, your program must tell us the final coordinates of the Robot.

Input to your program will be a series of directions (north, south, east, or west) each separated by one space.

The output to your program will be the final locatin of the Robot in Cartesian coordinates.

Sample

Input

```
west south east east east north north
```

Output

```
(2,1)
```

Palindromic Primes
(10 points)

Save in folder named: **PalPrimes**

A "palindromic prime" is a prime that is also a prime when its digits are reversed. 11, 13, and 17 are such primes.

Find and display the next ten "palindromic primes" that are not reversals of a smaller palindromic prime. Thus, 31 and 71 would not be displayed because they are the reversals of 13 and 17, which are already listed.

When you run out of two-digit numbers, switch to three, etc.

Perfect Numbers
(20 points)

Save in folder named: **Perfect**

The Greeks began an examination of numerology by classifying all positive integers as either **perfect**, **abundant**, or **deficient**. This classification scheme is based on the factors (even divisors) of the number. If the sum of all the factors of a number (excluding the number itself) equals the number then it is said to be **perfect**. For example, the factors of 6 are 1, 2, 3, and 6. Therefore, the number 6 is a **perfect** number. The total of the factors of 6 (excluding 6 itself) is 1 + 2 + 3 = 6.

An **abundant** number is one in which this sum of its factors (excluding the number itself) is greater than the number. An example of an **abundant** number is 12 because 1 + 2 + 3 + 4 + 6 = 16 > 12. All numbers that are neither **perfect** nor **abundant** are **deficient**.

This program accepts a sequence of integers, calssifies each as abundant, perfect, or deficient and displays the factors of the number. It terminates when given a input of zero.

For input, the user is prompted by "Specify a positive integer (0 to quit): ".

For output, display a line like "**This number is** *type* **(factors givein below).**" where *type* is either **perfect**, **abundant**, or **deficient**. On the following line, the factors of the input integer are shown.

Sample:

```
Specify a positive integer (0 to quit): 6
This number is perfect (factors given below).
1 2 3
Specify a positive integer (0 to quit): 12
This number is abundant (factors given below).
1 2 3 4 6
Specify a positive integer (0 to quit): 333
This number is deficient (factors given below).
1 3 9 37 111
Specify a positive integer (0 to quit): 0
```

Poly Wanna
(30 points)

Save in folder named: **PolyWanna**

You promp the user to enter values for the center and radius of a circle followed by the number of sides of a regular polygon. Then you graphically draw the circle with a regular polygon inscribed in the circle having the number of sides specified.

Here's a little help with the coding (which you'll only get here at the Heritage Computer Challenge). Start a GUI app, get rid of all the instance variables, and then add your own static instance variables. Have the user input these values in the main method before you instantiate your JFrame. The add a special method named `paint` like this:

```
public void paint(Graphics g)
{
    super.paint(g);
    g.drawLine(0,20,250,100);
    // remove the sample line drawing command and put your own stuff here
    // use online help for the Graphics class to figure out what to do
}
```

Sample run:

```
Poly Wanna ~ by Mr. C. Monroe
Enter x: 200
Enter y: 200
Enter radius: 150
How many sides to your polygon? 8
```

Resulting GUI output: