The Heritage Computer Challenge
2003
Heritage High School
Newport News, Virginia

C++

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 hcc2003. All other project folders will created **inside this folder.**
2. Each solution should be saved as a project in a folder whose name is IDENTICAL to the project file name (minus the extension). For each project, place your source code in the default source file named main.cpp.

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

1. Where is My Robot? (10 points)
2. Palindromic Primes (10 points)
3. Perfect Numbers (20 points)
4. Armadillo Air (20 points)
5. Text Formatting (30 points)

Where is My Robot?
(10 points)

Save in folder named: **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): ".

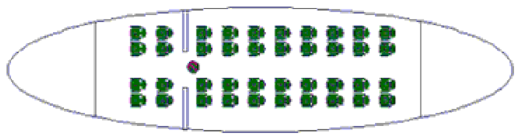
For output, display a line like "**This number is** *type* **(factors givein below).**" where *type* is either **perfect**, **abundant**, or **deficient**. On the following line, the factors of the input integer are shown.

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
```

Armadillo Air

(20 points)

Armadillo Air, a fledgling subsidiary of Auckland Airlines has initiated domestic flights from Norfolk to selected U.S. cities. No expense had been spared to guarantee that the first day of operations would be truly a historic event. Upon entering the terminal one was immediately surrounded by an overabundance of wine and exotic food from around the world. Adding to the festive atmosphere was the music provided by only the finest of bands. The fact that the Rolling Stones were booked as an opening act is an indication of the quality of performing artists. The celebration ceased when the Auckland executives arrived. They, because of their keen business sense, noticed certain problems that had been overlooked by the local staff. They could accept the fact that this new airline lacked flight attendants, pilots and even planes, but they insisted that all airlines under their control should have a reservation system. Your task is to create this reservation system.



Display a two dimension array to represent the seat locations. There should be four rows identified by the letters A, B, C, D, and ten columns identified by the digits 1 through 10. The following is provided as an example:

1D	2D	3D	4D	5D	6D	7D	8D	9D	10D
1C	2C	3C	4C	5C	6C	7C	8C	9C	10C
1B	2B	3B	4B	5B	6B	7B	8B	9B	10B
1A	2A	3A	4A	5A	6A	7A	8A	9A	10A

The passenger should be asked to enter their name, the number of seats desired and the seat location(s). After this has been entered for a passenger, the screen should be redisplayed with a special character or characters replacing the original seat identifiers. These seats are then occupied, and can not be reserved by future passengers.

Your program should allow up to five passengers to enter their information. When a passenger enters a seat location that has already been reserved this request should be rejected.

Text Formatting
(30 points)

Save in folder named: **Formatting**

The problem is to reformat a paragraph to a specified width. This means that no line in the paragraph should be longer than the width specified and each line should be as long as possible without going over the specified with. In the paragraph, words are separated by spaces and line breaks. No word should be split between two lines in the output. Leading or trailing punctuation should be treated as part of the word.

Input

Any number of lines of text, followed by a line which begins with an asterisk (*), followed by a space, followed by an integer. The integer specifies the width of the paragraph. The asterisk is not considered to be part of the paragraph.

Output

The input lines of text reformatted to the width specified.

Sample
Input
```
This function treats a string
(the value of EXPR) as a vector of unsigned integers, and
returns the value of the element specified by OFFSET and
BITS.
The function may also be assigned to, which causes
the element to be modified.
* 40
```

Output
```
This function treats a string (the value
of EXPR) as a vector of unsigned
integers, and returns the value of the
element specified by OFFSET and BITS.
The function may also be assigned to,
which causes the element to be modified.
```