

Programming Problems

13th Annual Computer Science Programming Contest

Department of Mathematics and Computer Science
Western Carolina University
April 23, 2002

Criteria for Determining Team Scores

Each program should:

1. execute without error, solve the appropriate problem efficiently, and satisfy all the conditions specified in the problem statement
2. follow good programming style conventions such as avoiding confusing and unnecessary code,
3. be documented with comments and easily understandable variable names, and
4. start with an initial comment that includes a) the name of the team's school, b) the team's number, and c) the names of the team's members. An example such comment is

// Somewhere High School, Team 2, Susan Smith, Shelley Jones, and Sandy Hart

The score assigned to each program will be based upon the extent that the program satisfies the properties listed above. The total accumulated score for all four programs represents the team score. All the programs have the same weight in the scoring.

Notes

- 1) Each team may use one C++ reference book.
- 2) You can use more than one computer if you wish.

Problem 1 (Degree of Inversion)

Compute the extent to which an array of integers is out of sorted order with sorted order meaning that the smallest index has the smallest value, the second index has the second smallest value, and so on until the largest index has the largest value. The metric for computing the degree of inversion is defined as follows. For each array index i count how many elements at larger indexes have values strictly smaller than the value of the element at index i . The degree of inversion is the sum of this count over all indexes. Notice that if the array is sorted, then the degree of inversion is 0. Your program output must look like the example output shown below. User input is shown in italics and program output is shown in boldface.

Example Program Run 1

Please enter the array values: *4 3 2 1*

The degree of inversion is: **6**

Example Program Run 2

Please enter the array values: *1 2 3 4*

The degree of inversion is: **0**

Problem 2 (Displaying an Integer with Commas)

Write a program that prompts the user for an integer. It then displays the integer (and an explanatory string) with commas between every set of three digits. Your program must have the following properties.

- The solution must be able to handle arbitrarily large integer values. You can not assume that the integer has at most some specific finite number of digits (such as nine). For example, if the maximum size of the int data type were to be increased on a new machine, your program should not fail.
- You must read in and process the integer as an integer, not as a string.
- Your program output must look like the example output shown below. User input is shown in italics and program output is shown in boldface.

Example Program Run 1

Please enter one integer: *123456789*
The integer with commas is **123,456,789**

Example Program Run 2

Please enter one integer: *12345*
The integer with commas is **12,345**

Example Program Run 3

Please enter one integer: *12*
The integer with commas is **12**

Problem 3 (Pattern Detection)

You are given two sequences of characters of length five. You are to see if the characters in the two sequences follow the same pattern. The definition of the *same pattern* is:

For each character, A, in the first sequence there must be a match character, B, with which every occurrence of A can be replaced. After all characters in the first sequence are replaced by their match characters, then the two sequences will be identical.

Your program output must look like the example output shown below. User input is shown in italics and program output is shown in boldface.

Example Program Run 1

Each sequence has five characters.

Please enter the first character sequence: *a a b b c*

Please enter the second character sequence: *x x y y z*

The two sequences follow the same pattern.

Example Program Run 2

Each sequence has five characters.

Please enter the first character sequence: *a a b b c*

Please enter the second character sequence: *x y y y z*

The two sequences do not follow the same pattern.

Example Program Run 3

Each sequence has five characters.

Please enter the first character sequence: *a a b b c*

Please enter the second character sequence: *x x y y x*

The two sequences do not follow the same pattern.

Problem 4 (Rock, Paper, Scissors)

Write a program that plays the game Rock–Paper–Scissors with the user until the user does not want to play any longer. In a single game the user makes a choice of rock, paper, or scissors and the computer does also. The computer's choice is made randomly. Rock beats scissors, scissors beats paper, and paper beats rock.

Your program must have the following properties.

- When your program prompts the user about playing another game, your program must handle the user entering an invalid value (that is, not *yes* or *no*) by detecting that the value is invalid and repeating the prompt until a valid answer is given by the user.
- When your program prompts the user about making a choice of *rock*, *paper*, or *scissors*, your program must handle the user entering an invalid value by detecting that the value is invalid and repeating the prompt until a valid answer is given by the user.
- The computer's choice must be generated randomly. In C++ the function to use is called *random()*. The function *random()* returns an integer between 0 and a large value, so you may want to use the mod operator to make the result either 0, 1, or 2. You can use *random()* in your program as long as your program has the preprocessor declaration *#include <stdlib.h>*.
- Your program output must look like the example output shown below (except that because the computer's choice is randomly generated, its choice may differ). User input is shown in italics and program output is shown in boldface.

Example Program Run 1

Do you want to play a game of Rock–Paper–Scissors (yes/no)? *maybe*

Your answer was not yes or no. Please try again: *yes*

Please enter your choice of rock, paper, or scissors: *rock*

Computer wins. user == rock; computer == paper

Do you want to play a game of Rock–Paper–Scissors (yes/no)? *yes*

Please enter your choice of rock, paper, or scissors: *rock*

Tie. user == rock; computer == rock

Do you want to play a game of Rock–Paper–Scissors (yes/no)? *yes*

Please enter your choice of rock, paper, or scissors: *stone*

Your choice was not rock, paper, or scissors. Please try again: *scissors*

User wins. user == scissors; computer == paper

Do you want to play a game of Rock–Paper–Scissors (yes/no)? *no*

Thanks for playing.