

CS 350-Fall 2005

Project 3 Requirements

v. 3/4/05

Purpose: Deal with programming issues that may be new to you. This assignment is an example of prototype code that might be written to determine how the “real” code in a project might be designed and written.

Goals: Use C++ to handle the following:

1. 8-bit integers,
2. Data represented in sign-magnitude rather than twos-complement notation,
3. Packing several types of data into an array (as is typical in a network data packet).

Discussion: The NASA spec requires you to write code that uses 8-bit integers, process data represented in sign-magnitude notations, and prepare data packets for transmission. This assignment isolates these issues so that they can be well understood before dealing with complexities in the context of a large project.

This assignment should also illustrate one aspect the Oracle problem since determining if your code is working correctly will likely require some mental effort.

Process steps: Thus the code is an example of building several small prototypes to check out some coding issues.

Requirement 1 (use 8-bit integers): Read in two integers and print their sum, difference (subtract the second from the first), and product. The integers must be unsigned and stored in 8-bit memory locations. You should demonstrate that you are using 8-bit integers by including at least one test case that causes an overflow (that is, the sum or product of the two integers is too large to fit into 8 bits).

Requirement 2 (convert sign-magnitude to twos-complement): Sum two integer values whose representation in memory is in 32-bit sign-magnitude format. Print the sum of the two values. Note: the problem here is that the hardware you are using likely does use sign-magnitude representation. Thus, to perform arithmetic operations using machine-supported arithmetic operations, you must convert the sign-magnitude representation to twos-complement representation prior to adding their values. In your testing, you should include a mixture of positive and negative input values to demonstrate that you are handling signs correctly and are converting correctly.

Hints & comments: you will be running your code on a hardware platform that uses two's complement representations for integers. This has nothing to do with the operating system you used, it is a characteristic of the hardware so how you handle this is the same whether running Windows, Mac OS 10, Linux, or Solaris.

The testing code can be written in several ways, but to simplify testing for the grader, you are must use the following steps:

- A. Get integers in sign-magnitude format in memory:
 1. Read two values using `cin` (which converts ASCII strings to twos-complement),
 2. Convert the twos-complement representations to sign-magnitude,
 3. Display, using `cout` the converted values (which will print a different value since the person who wrote `cout` assumed all integers would be in twos-complement format)
- B. Convert the sign-magnitude representations to twos-complement

- C. Perform the addition.
- D. Display the results of the addition.

The only purpose of steps A, C, and D is to test the code of step B; step B code will be needed for the term project.

Requirement 3 (pack different data types into an array): Declare an array of size 12 of short ints. Then read 6 numeric values into 6 variables of different types (listed below) and place those values into the specified array locations. The data types of the 5 variables must be as follows:

Position	C++ Type	Number Bytes
1	char	1
2	double	8
3	int	4
4	double	8
5	short	2
6	char	1

Then print the contents of the array as short ints.

Testing: A significant part of your effort will be to test these code fragments. To do so, you must determine, for each set of test data (6 numbers), what a correct program would print. To do this, you will likely need to review internal representations for some data types, then determine, for each bit pattern in an internal representation of some data type, if that bit pattern were treated as a short int, what its value would be. You must do this in order to be able to decide if the output produced by a program is correct.

Error handling: none required; this is just a prototype to confirm that you know how to handle some programming details that will be needed later.

Input: A file consisting of 10 numeric values (two for requirement 1, two for requirement 2, and 6 for requirement 3).

Output: A listing of 18 numeric values (3 from requirement 1, 3 from requirement 2, and 12 from requirement 3).

Additional Requirements:

All input should be read from standard input. All output should be written to standard output.

What Must Be Submitted:

1. Project Plan Summary, Initial (including time estimates for each phase as well as the regular size estimates).
2. Planned test data: a collection of test data files for each test case consisting of:
 - a. An input file which contains 10 numeric values
 - b. A statement of test objective for the test case.
 - c. A file containing expected output that a correct program would produce if it read the input file of a.
3. Source code and a supporting Makefile: As in previous assignments, do **not** include an executable.
4. Completed Project Plan Summary, Final: Note that the supporting documentation for this form (time log, etc.) are not requested this time.
5. Modified test data, in case you discovered errors in your test data or realized that additional test cases were useful.

6. Test logs. These logs provide, for each test case:
 - The time and date the testing was performed.
 - The name of the tester (you, in this case)
 - The results of the testing: For example, it might state that all results matched the expected results or that some faults were found. If so, describe them.
 - The descriptions should be short. You can combine the results of running several test cases into one report, but you should describe the results of each test separately.
7. A submission checklist (this list of items), indicating what from this list you have submitted, along with the date of each submission.

Due Dates: See document on class web site.