CSCI-1200 Computer Science II — Spring 2006
Test 2 — Practice Problems

General Information

- Test 2 will be held **Tuesday, March 7th, 2006 10-11:50am, West Hall Auditorium**. No make-ups will be given except for emergency situations, and even then a written excuse from the Dean of Students office will be required.
- Coverage: Lectures 1-12, Labs 1-7, HW 1-5.
- Closed-book and closed-notes *except for 1 sheet of 8.5x11 inch paper (front & back) that may be handwritten or printed.* Computers, cell-phones, palm pilots, calculators, PDAs, etc. are not permitted and must be turned off.
- All students must bring their Rensselaer photo ID card.
- Below are relevant sample questions from previous tests. The best thing you can do to prepare for this test is practice. Try these problems with pencil & paper first. Then practice programming (with a computer) these exercises and other exercises from lecture, lab, homework and the textbook.

Practice Problems

1. Write a code segment that copies the contents of a string into a list of char in reverse order.
2. Write a code segment that removes all occurrences of the letter 'c' from a string. Consider both uppercase 'C' and lowercase 'c'. For example, the string "`Chocolate`" would become the string "`hoolate`".
3. Write a function that rearranges a list of doubles so that all the negative values come before all the non-negative values AND the order of the negative values is preserved AND the order of the positive values is preserved. For example, if the list contains:

   ```
   -1.3, 5.2, 8.7, 0.0, -4.5, 7.8, -9.1, 3.5, 6.6
   ```

   Then the modified list should contain:

   ```
   -1.3, -4.5, -9.1, 5.2, 8.7, 0.0, 7.8, 3.5, 6.6
   ```

   This is a challenging problem, but it is good practice. Try to do it in two different ways: one without using an extra list and one using an extra list.

4. Write a function that determines if the letters in a string are in alphabetical order. Whitespace characters and punctuation characters in the string should be ignored in deciding if the letters are in alphabetical order. For example:

   ```
   b *((* B  Eee& &^E fg rz!!
   ```

   is in alphabetical order, but:

   ```
   b *((* B  Eee& &^Ea fg rz!!
   ```

   is not.

5. Consider the following start to the declaration of a `Course` class.

   ```
   class Course {
   public:
     Course(const string& id, unsigned int max_stu)
        : course_id(id), max_students(max_stu) {}



   private:
     string course_id;
     list<string> students;
     unsigned int max_students;
   };
   ```

   Use this in solving each of the following problems.

   (a) Provide three member functions: one returns the maximum number of students allowed in the `Course`, a second returns the number of students enrolled, and a third returns a `bool` indicating whether or not any openings remain in the `Course`. Provide both the prototype in the declaration above and the member function implementation.

   (b) Write a function that sorts course objects by increasing enrollment. In other words, the course having the fewest students should be first. If two courses have the same number of students, the course with the smaller maximum number of students allowed should be earlier in the sorted vector.

   (c) Write a member function of `Course` called `merge`. This function should take another `Course` object as an argument. All students should be removed from the passed `Course` and placed in the current course (the one on which the function is called). You may assume (just for this problem) that no students are enrolled in both courses and there is enough room in the current course.

   As an example if `cs2` is of type `Course` and has 15 students and `baskets` is of type `Course` and has 5 students, then after the call `cs2.merge(baskets);` `baskets` will have no students and `cs2` will have 20.

6. Write a function that takes a vector of strings as an argument and returns the number of vowels that appear in the string. A vowel is defined as an 'a', 'e', 'i', 'o' or 'u'. For example, if the vector contains the strings:

```
abe  lincoln  went  to  the  white  house
```

Your function should return the value 12. You may assume that all letters are lower case. Here is the function prototype:

```
int count_vowels(const vector<string>& strings)
```

7. Write a function that takes a list of doubles and copies its values into two lists of doubles, one containing only the negative numbers from the original list, the other containing only the positive numbers. Values that are 0 should not be in either list. For example, if the original list contains the values:

```
-1.3, 5.2, 8.7, -4.5, 0.0, 7.8, -9.1, 3.5, 6.6
```

then the resulting list of negative numbers should contain:

```
-1.3, -4.5, -9.1
```

and the resulting list of positive numbers should contain:

```
5.2, 8.7, 7.8, 3.5, 6.6
```

Start this problem by writing the function prototype as you think it should appear and then write the code.

8. Write a program that reads a sequence of strings from `cin` (up to the end of file) and determines the fraction of strings for which the vowels appear in alphabetical order. For example if the strings are:

```
Aardvark  birthday anybody count  car     peppermint
Ohio!! enough  re-iterate baseball
```

Then the output should be: 0.6 because there are 10 strings and 6 (`Aardvark`, `anybody`, `count`, `car`, `peppermint` and `enough`) contain their vowels in alphabetical order. You do not need to write down any `#include` statements, and you are welcome to write additional functions.

9. Write a function that merges two lists of float, each of which already in increasing order, to form a single list of floats. Try do this in two ways. In the first way, copy the values into a w vector. In the second, insert the values from one list into the her. Use the `list<T>::insert` function.

10. Write a function that finds the mode of a vector of integers. This is the integer that occurs most often. If two (or more) integers occur the same number of times then you should return the smallest. More computationally efficient solutions are better. For example, if the vector contains the values:

```
15, 45, 32, 16, 32, 83, 45, 89, 32, 15, 83
```

the function should return the value 32 because it occurs 3 times and no other value occurs more than twice.

11. Write a **recursive** function to add two non-negative integers using only a limited set of operations: adding 1 to a value, subtracting 1 from a value, and comparing a value to 0. No loops are allowed in the function. The function prototype should be:

```
int Add(int m, int n)
```

12. Write a **recursive** function to multiply two non-negative integers using only addition, subtraction and comparison operations. No loops are allowed in the function. The function prototype should be:

```
int Multiply(int m, int n)
```