CS 115 Final Review Quiz

December 11, 2008

Rules

- Reference material on the STL classes and algorithms, as well as functions that operate on C-strings, is included at the end of this document.

- You must briefly explain your answers to receive partial credit.

- When a snippet of code is given to you, you can assume that the code is enclosed within some function, even if no function definition is shown. You can also assume that the `main` function is properly defined and that the `iostream`, `fstream`, `iomanip`, `vector`, `algorithm`, `string`, `cstring`, and `cmath` libraries have been included at the beginning of the program.

- When you are asked to write *a snippet* of code, you may also assume that it is enclosed within some function that any necessary libraries have been included.

- When you are asked to write *a complete program*, you must write the `#include` statements, the `int main()`, etc. in your solution to receive full credit.

- A line consisting solely of "…" represents one or more unspecified C++ statements, some of which may change the values of program variables.

Problem 1: 15 points.

Match the following descriptions with the term they describe by writing that term in the space provided. The choices of terms are:

- this
- destructor
- class
- dynamic memory allocation
- overloading
- operator
- iterator
- container

Not all terms will be used.

(a) Multiple functions or operators with the same name but different input data types

(b) A function that is automatically called when an object is deleted

(c) The use of new or delete to create a variable or object

(d) A class, such as the STL's vector class, that stores a collection of objects.

(e) A pointer to the object whose member function is being called.

Problem 2: 10 points.

Correct the following program so that its behavior matches the initial comment.

```
/* Create a Student whose name is "Donald Duck" and whose
ID is 5050, and use the Print function in the Student class
to print the student's name and ID */
#include <iostream>

#include <string>

using namespace std;

class Student {

     string name;

     long id_num;

     Print() {
          cout << "Name: " << name << endl << "ID: "
               << id_num << endl;
     }
}

int main {

    Student.name = "Donald Duck";

    Student.id_num = 5050;

    Student.Print();

    return 0;
}
```

Problem 3: 10 points.

Write a *snippet* of code that does the following:

- Declares a vector of strings
- Adds three strings to the vector (your choice…keep it clean).
- Sorts the vector
- Empties the vector
- Prints the final size of the vector.

Remember to use the reference material at the end of this test.

Problem 4: 10 points.

(a) What's wrong with the following snippet of code? (You don't have to fix it; just clearly state what's wrong.)

```
/* Set C-string a equal to "pear" and b equal to "apple".
Then copy b to a and print out a. */
char* a;
char* b = "apple";
strcpy (a, b);
cout << a;
```

(b) Assume that the `Word` class has been declared with the following private data members (in addition to a bunch of public functions):

```
private:
     char * data; int length;
```

You are in charge of defining what the `*` operator means for this class; you decide that `w1 * w2`, where `w1` and `w2` are words, will return the product of the lengths of the "data" strings `w1` and `w2`. Someone has already written the skeleton for you; you just need to fill it in. For your reference, the definition of the + operator is also shown.

```
// Addition operator
Word Word::operator + (const Word& other) const {
   Word temp;
   temp.length = this->length + other.length;
   temp.data = new char [temp.length + 1];
   strcpy (temp.data, this->data);
   strcat (temp.data, other.data);
   return temp;
}
```

```
// Multiplication operator: fill this in
int Word::operator * (const Word& other) const {




}
```

Problem 5: 15 points.

For this problem, you must write a **class definition** for a class named **`Rectangle`** that contains the following:

- Length and width (as private data of type `double`)
- Prototype for a default constructor
- Prototype for a function called `SetSides`. This function will take two `doubles` as inputs and return a `bool`.
- Prototype for a function called `GetArea`. This function will return the area of the rectangle.
- Prototype for a function called `GetDiag`. This function will return the length of the rectangle's diagonal.

Problem 6: 20 points.

In this problem, you will write definitions for the functions in the class `Rectangle`. Here is a little bit more information about the functions.

- The default constructor will initialize the length and width to 1.
- The `SetSides` function will work as follows:
    - If one or both of the inputs is less than or equal to zero, it will return `false`.
    - Otherwise, it will set the length equal to the first input and the width equal to the second input. Then it will return `true`.
- The `GetArea` function will return the area of the rectangle (length times width).
- The `GetDiag` function will return the length of the rectangle's diagonal. Recall that the length, width, and diagonal form a right triangle that obey the relationship $a^2 + b^2 = c^2$, where a and b are the length and width and c is the diagonal. Assume that `<cmath>` is included.

Problem 7: 20 points.

Assume that the class definitions you wrote in Problems 5 and 6 are located in a file called `rectangle.h` in the same directory as the program you're about to write.

Write **a complete program** that uses the `Rectangle` class from `rectangle.h` to do the following:

- Create one `Rectangle` object with `length` 5.5 and `width` 3
- Ask the user to supply the length and width of another rectangle, and create a `Rectangle` object for that one too. You can assume that the user enters a valid length and width.
- Use the `GetArea` function to compute the area of both rectangles, and print ONLY the area of the larger one.

REFERENCE

C-string functions:

`strlen`	Input is a C-string. Returns the length of the string (not including the null terminator). Usage example: `length = strlen(name);`
`strcat`	Input is two C-strings. Appends the second string to the end of the first string (the first string is changed, but the second is not). Usage example: `strcat(str1, str2);`
`strcpy`	Input is two C-strings. Copies the second string to the first string, overwriting the original contents. Usage example: `strcpy(str1, str2);`
`strcmp`	Input is two C-strings. Returns 0 if they are the same, a negative number if `str2` is alphabetically greater than `str1`, and a positive number if `str1` is greater than `str2`. Usage example: `if(strcmp(str1, str2) > 0)`

Vector member functions:

`begin()`	Returns an iterator to the vector's first element
`clear()`	Removes all elements from the vector
`empty()`	Returns a bool which is true if the vector is empty (has zero elements) and false otherwise
`end()`	Returns an iterator pointing just past the vector's last element
`pop_back()`	Removes the last element from the vector
`push_back(value)`	Inserts value as a new element at the end of the vector
`size()`	Returns the number of elements in the vector

STL algorithms (NOT member functions)
Here *iter1* and *iter2* are iterators pointing to elements of an STL class such as `vector`.

`binary_search(iter1, iter2, value)`	Returns `true` if `value` is found in the range between `iter1` and `iter2`, `false` otherwise
`count(iter1, iter2, value)`	Returns the number of times `value` appears in the range between `iter1` and `iter2`
`reverse(iter1, iter2)`	Reverses the order of the elements between `iter1` and `iter2`
`random_shuffle(iter1, iter2)`	Randomly changes the order of the elements between `iter1` and `iter2`
`sort(iter1, iter2)`	Sorts the elements in the range between `iter1` and `iter2` in ascending order