# CS 115 Final Exam, Spring 2009

## Your name: _____

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

### Grade (instructor use only)

| | Score | Max |
|---|---|---|
| Problem 1 | | 15 |
| Problem 2 | | 15 |
| Problem 3 | | 15 |
| Problem 4 | | 15 |
| Problem 5 | | 20 |
| Problem 6 | | 20 |
| **Total** | | 100 |

## 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
- constructor
- destructor
- class
- private
- public
- object
- pointer
- STL
- dereference
- pass by reference
- pass by value

Not all terms will be used.

(a) Term for a member of a class that cannot be accessed from outside the class

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

(c) A way of allowing a function to modify its input arguments

(d) A programmer-defined C++ data type that can contain its own variables, functions, and operators

(e) The operation of following a pointer to the thing it points to (*i.e.* the operation `*(ptr)`)

## Problem 2: 15 points.

Using the selection sort code on the next page, which works identically to the code from your labs, show the contents of the following 5-element array after each iteration of `SelectionSort`'s loop in the labeled boxes. *Leave the boxes blank if the loop does not complete a given iteration.*

**INITIAL VALUE**

| | | | | |
|---|---|---|---|---|
| 8 | 6 | 1 | 2 | 3 |

**AFTER i=0**

| | | | | |
|---|---|---|---|---|
| | | | | |

**AFTER i=1**

| | | | | |
|---|---|---|---|---|
| | | | | |

**AFTER i=2**

| | | | | |
|---|---|---|---|---|
| | | | | |

**AFTER i=3**

| | | | | |
|---|---|---|---|---|
| | | | | |

**AFTER i=4**

| | | | | |
|---|---|---|---|---|
| | | | | |

**AFTER i=5**

| | | | | |
|---|---|---|---|---|
| | | | | |

**AFTER i=6**

| | | | | |
|---|---|---|---|---|
| | | | | |

## Code for Problem 2.

*You may tear this page out of your exam.*

```
void SelectionSort(int arr[], int size) {
   int min_pos = 0;

   // This is the loop in question
   for (int i = 0; i < size-1; i++) {
      min_pos = FindMinPos(arr, i, size);
      if (min_pos != i) {
          swap ( arr[i], arr[min_pos] );
      }
   }
}

int  FindMinPos(int arr[], int size) {

   int minVal = arr[0];
   int minPos = 0;

   for (int i = 1; i < size; i++) {
     if (arr[i] < minVal) {
          minVal = arr[i];
          minPos = i;
      }
   }
   return minPos;
}

void Swap (int& a, int& b) {
   int temp = a;
   a = b;
   b = temp;
}
```

## Problem 3: 15 points.

Complete the definitions of the following functions as indicated by their comments. You may assume that any necessary libraries have been included.

*Hint*: Each function can be written in two lines of code or fewer, none of which are `cin` or `cout` statements.

(A)

```
/* Sort the input vector and return the value of its
   smallest element. */
int Funct1 (vector<int>& s) {









}
```

(B)

```
/* Return the sum of the lengths of the two input
   C-strings. */
int Funct2 (char* a, char* b) {









}
```

(C) For this part, assume that the class `Rectangle` has private fields

```
int length;
int width;
```

You will be writing the > operator.

For your reference, the copy constructor for the class is given below:

```
/* Copy constructor for Rectangles. */
Rectangle::Rectangle (const Rectangle& other) {
     this->length = other.length;
     this->width = other.width;
}
```

```
/* Overloaded ">" operator for the Rectangle class.
   Returns TRUE if the area of this is greater than the
   area of other and FALSE otherwise.
   You fill this in. */
bool Rectangle:: operator > (const Rectangle& other) const{









}
```

## Problem 4: 15 points.

For this problem, you must write a **class definition** for a class named **`RT`** (as in "right triangle") that contains the following:

Note that data members should be private and member functions should be public.

- Variables for the three sides of the triangle (as `double` variables named `s1`, `s2`, and `hypo`)
- Prototype for a default constructor
- Prototype for a function called `SetSides`. This function will take **two** variables of type `double` as inputs and will return a `bool`.
- Prototype for a function called `GetHypo`. This function will return the length of the triangle's hypotenuse.
- Prototype for a function called `GetArea`. This function will return the area of the triangle.

## Problem 5: 20 points.

In this problem, you will write definitions for the functions in the class `RT`. Here is a little bit more information about the functions. You may assume that `<cmath>` is included. *Note that none of your code for this problem should include* `cin` *or* `cout` *statements!*

- The default constructor will initialize the three side lengths to 0.
- The `SetSides` function will work as follows:
  - If one or both of the inputs is less than zero, it will return `false`.
  - Otherwise:
    - It will set `s1` equal to the first input.
    - It will `s2` equal to the second input.
    - It will set `hypo` equal to the square root of ($s1^2$ + $s2^2$). Then it will return `true`.
- The `GetHypo` function will return the length of the hypotenuse.
- The `GetArea` function will return the area of the triangle, which is half of the product of `s1` and `s2`.

## Problem 6: 20 points.

Assume that the class definition you wrote in Problems 4 and 5 are located in a file called `rt.h` in the same directory as the program you're about to write.

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

- Create one `RT` object with the lengths of the two shorter sides set to 3 and 4.
- Create another `RT` object and ask the user to supply the lengths of the two shorter sides.
- Using `RT`'s member functions:
    - Print the hypotenuse of the user's triangle.
    - Print "The areas are equal" if the areas of the two triangles are equal.

## REFERENCE

**C-string functions:**

| | |
|---|---|
| `strlen` | Input is a C-string. Returns the length of the string (not including the null terminator).<br>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).<br>Usage example: `strcat(str1, str2);` |
| `strcpy` | Input is two C-strings. Copies the second string to the first string, overwriting the original contents.<br>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`.<br>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 |