CS 115 Midterm 2 Solutions

April 9, 2009

Rules

- 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`, `string`, 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.
- You are encouraged to use the backs of these pages for scratch paper. If you want answers written there to be graded, they must be very clearly labeled and also noted on the main test, e.g. "See the back of page 1 for 3a."

Problem 1: 15 points.

What is the output of each of the following snippets of code?

(a) If the following function is defined somewhere in the program and prototyped above main....

```
int my_function (int x, float array[]) {
        array[1] = 2.5;
        return x*2;
}
```

...what does the following code print?

```
int x = 5;
float a[3] = {0.2, 0.4, 0.6};
int y = my_function(x, a);
cout << x << endl;
cout << y << endl;
cout << a[1] << endl;
cout << a[2] << endl;
```

Answer:
5
10
2.5
0.6

(b)

```
for (int i=0; i < 2; i++) {
   for (int j=1; j < 3; j++) {
        cout << "i=" << i;
        cout << ", j=" << j << endl;
   }
}
```

Answer:
i=0, j=1
i=0, j=2
i=1, j=1
i=1, j=2

Problem 2: 25 points.

The snippets of code in this problem do not successfully accomplish the task described in their accompanying comments. Correct the code so that it performs the task described in the comment. Each snippet may have more than one error. **Make your corrections clear and unambiguous.**

(a)

```
/* Function that squares every element in an array of
floats. Inputs are the array and its size. No return
value.*/
float Square(float[] array) {

     for (i=0; i != size; i++) {
          array[i] = i^2;
     }

}
```

Corrections:

- Return type should be void
- Brackets in function header should be moved: float array[]
- int size should be included as a function input
- The loop counter i should be declared as an int
- The loop test i != size should be i < size
- We want to square each element of the array, and not the subscript i,, plus the ^ is not how we raise a number to a power. Should be array[i] = array[i] * array[i];

Corrected code:

```
void Square(float array[]) {
     for (int i=0; i < size; i++) {
          array[i] = array[i] * array[i];
     }
}
```

(b)

```
/* Defines a struct named Student to hold a student's
name, address, student ID number, and email  */
struct {
     string Student.name,
     string Student.address,
     int Student.id,
     string Student.email
}
```

Corrections:

- The struct needs a name
- Each field should have a semicolon at the end, rather than a comma
- Shouldn't use the "." notation when we're defining the struct
- Definition needs a semicolon at end

Corrected code:

```
struct Student {
      string name;
      string address;
      int    id;
      string email;
};
```

(c) Assume that the 2D array

```
int x[100][5]
```

has already been declared and defined.

```
/* Prints out all the elements of the 2D array X */
for (int i=0; i<=5; i++) {
     cout X[i][j];
}
```

Corrections:

- Loop test should be i < 5, not i <=5
- Need an outer loop that starts at 0 and stops at 99
- Need to print X[j][i]
- Need arrows in cout

Corrected code:

```
for (int j=0; j< 100; j++) {
     for (int i=0; i<=5; i++) {
          cout << X[j][i];
     }
}
```

Problem 3: 30 points.

Write short snippets of code to accomplish the following tasks:

(a) For an array that has been declared as

```
float floatArr[5][8];
```

write a snippet of code that computes and prints the total of all of the elements in the **first column** of the array.

Solution:

```
float sum = 0;
for (int i=0; i<8; i++) {
     sum += floatArr[0][i];
}
cout << sum << endl;
```

(b) Write a function called `AllEqual` whose input is an integer array of size 10. The function should return a `bool` whose value is `true` if all of the elements of the array are equal and `false` otherwise.

Solution:

```
bool AllEqual(int array[]) {
        int element = array[0];
        for (int i=1; i< 10; i++) {
                if (array[i] != element) {
                        return false;
                }
        }
        return true;
}
```

(c) Write a snippet of code that
- Declares an integer array of size 10 and initializes every element to the value 0
- Calls `AllEqual` (your function from part (b)) with this array as input
- Prints the value returned by the function call.

Solution:

```
int array[10];
for (int i=0; i<10; i++) {
        array[i] = 0;
}
cout << AllEqual(array);
```

(d) Write a snippet of code that defines a `struct` called `Person` with fields for a person's name and age. These fields can be any reasonable data type.

Solution:

```
struct Person {
        string name;
        int age;
};
```

(e) Write a snippet of code that declares a variable of type `Person` (based on your code from part (d)) and initializes the name to your name and the age to whatever (plausible) age you like.

Solution:

```
Person p;
p.name = "Suzanne Rivoire";
p.age = 21;              // I wish
```

(f) Write a snippet of code that defines a `struct` called `FootballTeam` with fields for:
- The name of the team
- The number of players currently on the team
- Up to 70 players, each of type `Person` (see parts (e) and (f))

Solution:

```
struct FootballTeam {
        string name;
        int num_players;
        Person players[70];
};
```

Problem 4: 30 points.

For this problem, you must write a **complete program** that contains the following:

- A definition for the const `NUM_SKATERS`, an integer equal to 80

- The definition of a `struct` called `Skater` that has two fields. You should decide the appropriate data types for these fields:
  - `name`, the name of a figure skater
  - `score`, the skater's latest score, which could be fractional

- Prototype for a function named `GetSkaterInfo`, which
  - Requires no input
  - Returns a `Skater`

- The definition of `GetSkaterInfo`. `GetSkaterInfo` should prompt the ***user*** to type in:
  - The name of a `Skater`
  - A score for that `Skater`

  The function should return a `Skater` with the name and score supplied by the user.

- A `main` function that does the following:
  - Declares **an array** of `NUM_SKATERS Skater` variables
  - In a loop, calls `GetSkaterInfo` to fill in each of the elements of the array
  - Prints the name and score of the first skater in the array

[see next page]

```
#include <iostream>
using namespace std;

const int NUM_SKATERS = 80;

struct Skater {
	string name;
	float score;
};

Skater GetSkaterInfo();

int main() {

	Skater s[NUM_SKATERS];

	for (int i=0; i<NUM_SKATERS; i++) {
		s[i] = GetSkaterInfo();
	}
	cout << s[0].name << endl;
	cout << s[0].score << endl;

	return 0;
}

Skater GetSkaterInfo() {
	Skater sk;
	cout << "Name?" << endl;
	cin >> sk.name;
	cout << "Score?" << endl;
	cin >> sk.score;
	return sk;
}
```