CS 101, Spring 2016 — May 5th — Exam 4 Name: Solution

For Questions 1–5, circle **True** or **False**.

Question 1. [2 points] **True** **False** A structure is a user-defined data type.

Question 2. [2 points] **True** **False** By default, structures are *passed-by-reference*.

Question 3. [2 points] **True** **False** By default, an array of structures is *passed-by-reference*.

Question 4. [2 points] **True** **False** A function may return multiple structure variables simultaneously through its return statement.

Question 5. [2 points] **True** **False** If `Point` is a struct type, the following array declaration is legal:

```
struct Point myPoints[10];
```

Question 6. [5 points] Which of the following can be included in a `struct` declaration (circle *all* that apply):

a. `int` variables

b. `double` variables

c. array of `int`'s

d. `struct` variable

e. `void` function declarations

f. pointers to `struct`'s

g. array of `struct`'s

Question 7. [10 points] Consider a structure named `Student` to store student information.

a.[4] Declare the `Student` struct type such that it contains fields to store the a student's age, GPA, gender (either `'M'` or `'F'`), and number of credits earned. Use appropriate data types and meaningful variable names for each.

```
struct Student {
    int age;
    double gpa;
    char gender;
    int credits;
};
```

b.[3] Declare a variable of type `struct Student` and assign values of your choice to the member fields.

```
struct Student s;
s.age = 19;
s.gpa = 3.71;
s.gender = 'F';
s.credits = 16;
```

c.[3] Write a function *prototype* for a function named `print_student_info()` that takes a `struct Student` as a parameter *by reference*.

```
void print_student_info(struct Student *p);
```

Question 8. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

struct Box {
   int side;
};

struct Box halveIt(struct Box b) {
  b.side = b.side / 2;
  return b;
}
```

updated value is returned

```
int main(void) {
  struct Box mine;
  mine.side = 42;
  mine = halveIt(mine);
  printf("%i\n", mine.side);
  return 0;
}
```

value returned by halveIt function is assigned to mine

a. 0

b. 21

c. 42

d. The output can't be predicted

Question 9. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>
struct Stuff {
  int a;
  float b;
};

struct Stuff dblStuff(float x, int y) {
  struct Stuff s;
  s.a = y;
  s.b = 2*x;
  return s;
}
```

4.0 2

// assigns 2

// assigns 8.0

value of s is returned

```
int main(void) {
  struct Stuff oreo;
  int cookie = 2;
  float filling = 4;
  oreo = dblStuff(filling,cookie);
  printf("%i %.1f\n",oreo.a, oreo.b);
  return 0;
}
```

4.0 2

2 8.0

value returned by function is assigned to oreo

a. 2 4.0

b. 4 4.0

c. 2 8.0

d. 8.0 2

e. 4.0 4

Question 10. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

struct Stuff {
  int hq;
  int lq;
};

struct Player {
   struct Stuff s;
   int hp;
};

int getStuff(struct Stuff *stuff,
             int a, int b) {
  stuff->hq += a;
  stuff->lq = stuff->hq - b;
  return stuff->lq;
}
```

```
int main(void)
{
  struct Player zelda;
  zelda.s.hq = 0;
  zelda.s.lq = 2;
  zelda.hp = 10;

  zelda.hp = getStuff(&(zelda.s),
                      zelda.hp, 1);

  printf("%i %i %i\n", zelda.s.hq,
                       zelda.s.lq,
                       zelda.hp);

  return 0;
}
```

zelda: s { hq 0, lq 2 }, hp 10 — initial contents

hq increases by 10 (to 10)

lq is set to 9

returns 9

zelda.hp = getStuff(&(zelda.s), zelda.hp, 1); — set to 9; 10

zelda: s { hq 10, lq 9 }, hp 9 — after call to getStuff

output is: 10 9 9

Question 11. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

struct Play {
   int score;
   int bonus;
};

 void calculate(struct Play *p,
                int n) {
  p->score++;
  p->bonus += n;
}
```

```
int main(void) {
  struct Play pl;
  pl.score = 10;
  pl.bonus = 15;
  calculate(&pl,5);
  printf("%i %i\n", pl.score
                    pl.bonus);

  return 0;
}
```

pl: score 10, bonus 15 — initial contents

increases by 1 to 11

5

increases by 5 to 20

pl: score 11, bonus 20 — after call to calculate

output is: 11 20

Programming Questions

Note: For all of the programming questions, you should use `scanf` to read the input value(s) required by the program.

Note: Make sure your programs produce the output in **exactly** the format described, including capitalization and punctuation. You may not receive credit for programs that produce incorrectly-formatted output.

Getting started: Start **Cygwin Terminal** and **Notepad++** and make sure ALL TABS are closed. (Note: do *not* open any other programs.) Your instructor will give you the name of a zip file. In your terminal, run the following commands:

```
cd h:
mkdir -p CS101
cd CS101
curl -O http://faculty.ycp.edu/~dbabcock/spring2016/cs101/zipfile
unzip zipfile
cd CS101_Exam4
```

Note that in the `curl` command, the `-O` has the letter 'O', not the digit '0'.

Substitute the name of the zip file for *zipfile*.

Editing code: Use your text editor to open the source file (e.g., `question12.cpp`) referred to in the question. Do not open any files other than the ones for the exam.

Compiling: To compile the program for Question 12, run the following command in the terminal:

```
make question12.exe
```

Change the number as appropriate for the other questions (e.g., `question13.exe`).

Running: To run the program for Question 12, run the following command in the terminal:

```
./question12.exe
```

Change the number as appropriate for the other questions (e.g., `question13.exe`).

To submit: In Cygwin Terminal, run the command

```
make submit
```

Enter your Marmoset username and password when prompted.

Good luck!

Question 12. [10 points] Finish writing the program in the source file `question12.cpp`. The program prompts the user for two integer values that represent the x and y coordinates of a point; these values are stored in variables `user_x` and `user_y`. You will modify the program so that

1. the values entered (`user_x` and `user_y`) are passed to the `init_Point` function, and
2. the return value of `init_Point` is assigned to the `struct Point` variable called `p`

The function prototype for the `init_Point` function has been provided for you. After the variable `p` has been initialized with the values entered by the user, the program prints the values stored in the `Point struct`.

Your task is to complete the program according to the following specification:

- you **must** write the `init_Point` function and it must match the provided function prototype
- **do NOT** modify the included `printf` statements

An example run of the program is below (user input in bold):

```
Enter x and y values for a Point: 22 33
Point: x=22, y=33
```

Question 13. [15 points] Complete the program `question13.cpp` as follows.

Code is provided in the program's `main` function to prompt the user to enter three `double` values storing them in the fields of a `struct Values` variable named `num` and output print statements. **DO NOT MODIFY** these parts of the program in any way.

Your task is to add the declaration for the `struct Values` type, a function prototype and definition for the `processValues()` function, and an appropriate call to this function in `main()`. Code should only be added in the indicated locations.

Part I

Add a declaration for a `struct Values` datatype above `main()` that contains three `double` fields named `x`, `y`, and `scale`.

Part II

Add the function definition (below `main()`) for a function named `processValues()` with a *pointer* to a `struct Values` parameter and that returns a `double` as shown in the function prototype. The function should multiply the `x` and `y` fields of the parameter by the `scale` field and return the larger of the scaled `x` and `y` fields.

Part III

Add a function call to `processValues()` that passes a *reference* to the `num` variable and stores the return value in the variable `ans`.

Example run (user input in bold):

```
Enter x/y: 3.0 4.5
Scale factor: 2.2
x=6.600000, y=9.900000
max=9.900000
```

Question 14. [15 points] Complete the program in `question14.cpp` as follows.

Two struct declarations are given for `struct Point` and `struct Circle`. The program's `main()` function (provided) reads `double` values for the center coordinates and the radii for two circles and loads them into corresponding `struct` variables `c1` and `c2`. It then calls the `does_intersect()` function, and prints out whether or not the circles intersect. **DO NOT MODIFY** `main()` in any way.

Your tasks are to add function definitions for the `compute_distance()` and `does_intersect()` functions. Function prototypes are provided at the top of the code.

- `compute_distance()` takes two `struct Point` parameters. The function should compute the distance between the two points using the formula $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ and return this distance as a `double` value. This function **should not** print any output.
- `does_intersect()` takes two `struct Circle` parameters *by reference*. This function should compute the distance between the centers **using the** `compute_distance()` **function** and print the result as shown in the example output. It should then compute the sum of the radii and print the result as shown in the example output. Finally the function should determine if the two circles intersect and return 1 if they intersect and 0 otherwise. Two circles intersect if the distance between their centers is less than the sum of their radii.

`does_intersect()` **MUST** call the `compute_distance()` function in order to receive full credit for this problem.

Example run (user input in **bold**):

```
Enter x, y, and r values for circle 1: 1.5 2.5 3.0
Enter x, y, and r values for circle 2: -1.0 2.0 4.25
The distance between the centers is 2.55
The sum of the radii is 7.25
The circles intersect
```

Another example run (user input in **bold**):

```
Enter x, y, and r values for circle 1: 0.75 1.2 1.0
Enter x, y, and r values for circle 2: -2.5 -4.0 2.1
The distance between the centers is 6.13
The sum of the radii is 3.10
The circles do not intersect
```

Hints:

- You **DO NOT** need to get any user input as that code has been provide.
- Function prototypes for the `compute_distance()` and `does_intersect()` functions are provided.
- To compute the square root use the `sqrt()` function.
- Don't forget to include a `return` statement at the end of each function.

Question 15. [15 points] Complete the program in `question15.cpp` as follows.

A struct declaration is given for `struct Employee`. The program's `main()` function (provided) reads an `integer` value for the number of employee weekly salaries to be entered. It then uses a `for` loop to read the `double` weekly salary values, and initializes each employee in the `employees` array by calling `initialize_employee()`. The `for` loop also has an initial `printf` statement which will be modified to output each element of the `employees` array to the console. `main()` then calls `calc_average_salary()`, which returns the average annual salary of all employees and prints the result.

Your tasks are as follows:

- TODO 1: Add the function prototypes for two functions. `initialize_employee()` which takes a `struct Employee` parameter *by reference*, an `integer` as the employee ID, and a `float` as the *weekly salary* for that employee. The function does NOT return a value. `calc_average_salary()` takes *an array* of `Employee structs` and the number of employees, i.e. the size of the array, and returns a `double` value.
- TODO 2: Call `initialize_employee()` inside the `for` loop for each employee's weekly salary that is entered. Note that employee IDs should start at 1 (not 0).
- TODO 3: Modify the `printf` in the `for` loop by replacing the constant values with appropriate variables to output each employee's information.
- TODO 4: Call `calc_average_salary()` to get the average annual salary of all the employees.
- TODO 5: Implement `initialize_employee()`. You must calculate the *annual salary* from the *weekly salary* that was read from the input, note there are 52 weeks in a year. You must also set the employee's tax rate, based on their annual salary. The tax brackets and tax rates are defined at the top of the file.
- TODO 6: Implement `calc_average_salary()` to compute the average salary of all the employees.

Example run (user input in **bold**):

```
Enter # of employees (max 10): 4
Enter employee 1 weekly salary: 500
Employee 1: $26000.00 (0.15)
Enter employee 2 weekly salary: 1000
Employee 2: $52000.00 (0.25)
Enter employee 3 weekly salary: 2000
Employee 3: $104000.00 (0.25)
Enter employee 4 weekly salary: 4000
Employee 4: $208000.00 (0.33)

Average salary: $97500.00
```

Hints (continued on next page):

- You **DO NOT** need to get any user input as that code has been provided.

- Additional information on implementing the two functions is included in the TODO comments for each function.
- Recall that reference parameters are specified as *pointers* in the function prototypes and function definitions (using `*`) , and that you must pass an *address* of a variable in the function call (using `&`).