CS 101, Spring 2013 — April 16th — Exam 3 Name: _____

Question 1. [3 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

void halveIt(int x) {
  x = x / 2;
}
```

```
int main(void) {
  int y = 8;
  halveIt(y);
  printf("%i\n", y);
  return 0;
}
```

A. 4

B. 8

C. The output can't be predicted

D. The program does not compile

Question 2. [3 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

void halveThem(int a[], int n) {
  for (int i = 0; i < n; i++) {
    a[i] = a[i] / 2;
  }
}
```

```
int main(void) {
  int nums[4] = { 2, 4, 6, 8 };
  halveThem(nums, 4);
  for (int i = 0; i < 4; i++) {
    printf("%i ", nums[i]);
  }
  return 0;
}
```

A. 1 2 3 4

B. 2 4 6 8

C. The output can't be predicted

D. The program does not compile

Question 3. [3 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 = b / 2;
  return b;
}
```

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

A. 21

B. 42

C. The output can't be predicted

D. The program does not compile

Question 4. [3 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 doubleStuff(float x,
                         int y) {
  struct Stuff s;
  s.a = y;
  s.b = 2*x;
  return s;
}
```

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

A. 2 4.0

B. 4 4.0

C. 2 8.0

D. The program does not compile

Question 5. [6 points] The following program reads a single integer value (n). Complete the program so that the `printf` statement in `main` prints the value 2^n (2 raised to the power n). You can assume n will be non-negative. You must:

1. Add a call to `multiplyByTwo` to the `for` loop in `main`
2. Complete the definition of the `multiplyByTwo` function

```
#include <stdio.h>

void multiplyByTwo(int *x);

int main(void) {
  int n;
  scanf("%i", &n);

  int product = 1;
  for (int i = 1; i <= n; i++) {









  }

  printf("%i\n", product);
  return 0;
}

void multiplyByTwo(int *x) {




}
```

Question 6. [3 points] Create a struct type called `Student` that has member fields to store the a student's age, GPA, sex (either `'M'` or `'F'`), and number of credits earned. Use appropriate data types and meaningful variable names for each.

Question 7. [3 points] Declare a variable of type `Student` (defined in Question 6) and assign values of your choice to the member fields.

Question 8. [4 points] Define a function called `print_student_info()` that takes a `struct Student` as a parameter and prints the student's age, GPA, sex, and number of credits earned in the following format:

age`,`*GPA*`,`*sex*`,`*numberOfCredits*

For example, for a 20-year old student, the output might be

```
20,3.71,F,45
```

For Questions 9–14, circle True or False.

Question 9. [2 points] True or False: It is possible to return an array from a function as a return value.

Question 10. [2 points] True or False: If `a` is an array parameter, it is possible to find out how many elements `a` has using the syntax `a.length`.

Question 11. [2 points] True or False: Structs allow you to define new data types.

Question 12. [2 points] True or False: It is permissible to assign the value of one struct variable to another struct variable, as long as the variables have the same types.

Question 13. [2 points] True or False: The ampersand (&) is the "address of" operator.

Question 14. [2 points] True or False: The asterisk (*) is the "address of" operator.

Programming Questions

Note: For all of the programming questions, you should **not** need to write any code to obtain user input.

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++** closing any open tabs in **Notepad++**. (Note: Do *not* open **ANY** other programs.) Your instructor will give you the name of a zip file. In Cygwin Terminal, run the following commands:

```
cd h:
mkdir -p CS101
cd CS101
curl -O http://faculty.ycp.edu/~dhovemey/spring2013/cs101/assign/zipfile
unzip zipfile
cd CS101_Exam3
```

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

Editing code: Use Notepad++ to open the source file (e.g., `question15.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 15, run the following command in Cygwin Terminal:

```
make question15.exe
```

Change the number as appropriate for the other questions (e.g., `question16.exe`).

Running: To run the program for Question 15, run the following command in Cygwin Terminal:

```
./question15.exe
```

Change the number as appropriate for the other questions (e.g., `question16.exe`).

To submit: In Cygwin Terminal, run the command

```
make submit
```

Enter your Marmoset username and password when prompted.

Good luck!

Question 15. [10 points] Finish writing the program in the source file `question15.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 16. [15 points] Complete the program `question16.cpp` as follows. The program reads three `double` values. The first two values are x and y coordinates. The third value is an arbitrary scaling factor. The program should multiply both the x and y values by the scaling factor and print out the resulting values. Example run (user input in bold):

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

Your task is to add a prototype and definition for the `scale_xy` function, which is responsible for multiplying the `user_x` and `user_y` variables by `scale_factor` and storing the updated results. Note that this function must use pointers as reference parameters so that the function call

```
scale_xy(&user_x, &user_y, scale_factor);
```

performs the computation *and* updates the variables.

Important: Do *not* change any of the code in the `main` function.

Hint: Only the first two parameters should be pointers (reference parameters).

Question 17. [15 points] Finish writing the program in the source file `question17.cpp`. The program prompts the user for an integer value representing the length of an array (up to a maximum of 10) and then inputs the integer array values. The program will then call a function named `arrayExtents()` which sets the first element of the array to its minimum and the last element to its maximum and then prints these extents.

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

- you **must** write the `arrayExtents()` function and function prototype which has an integer array parameter and an integer length parameter
- the function should *swap* the *first* element in the array with the *minimum*, i.e. smallest, element in the array
- the function should *swap* the *last* element in the array with the *maximum*, i.e. largest, element in the array
- **do NOT** modify `main()` where all output is done

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

```
Enter array length: 5
Enter array values: 4 1 5 2 3
1 4 3 2 5
```

Question 18. [20 points] Complete the program in `question18.cpp` as described below. The `struct Rect` and `struct Point` data types are defined as follows:

```
struct Rect {
  int xmin;
  int ymin;
  int xmax;
  int ymax;
};
```

```
struct Point {
  int x;
  int y;
};
```

An instance of the `struct Rect` data type represents a rectangle, where the `xmin` and `ymin` fields represent the minimum x and y coordinates, and the `xmax` and `ymax` fields represent the maximum x and y coordinates. The input for the rectangle will be of the form

Enter the rectangle bounds: `xmin xmax ymin ymax`

The `struct Point` data type represents a point, where the `x` and `y` fields represent the x and y coordinates. The input for the point will be of the form

Enter the point coordinates: `x y`

Your task is to write a prototype and definition for a `check_collision()` function, which takes a `struct Rect` and a `struct Point` as parameters. It should return an integer value as follows:

- If the point is outside of the rectangle, it should return 0
- If the point is inside the rectangle, it should return 1
- If the point is on the *top* edge of the rectangle, it should return 2
- If the point is on the *bottom* edge of the rectangle, it should return 3
- If the point is on the *left* edge of the rectangle, it should return 4
- If the point is on the *right* edge of the rectangle, it should return 5

The point is considered outside the rectangle if its `x` or `y` values are strictly less than `xmin` or `ymin`, *or* strictly greater than `xmax` or `ymax`.

The point is considered inside the rectangle if its `x` value is greater than `xmin` and less than `xmax`, *and* its `y` value is greater than `ymin` and less than `ymax`.

The point is considered on the top (bottom) edge if its `y` value is equal to `ymin` (`ymax`), *and* its `x` value is between `xmin` and `xmax`. Similarly the point is considered on the left (right) edge if its `x` value is equal to `xmin` (`xmax`), *and* its `y` value is between `ymin` and `ymax`. For corner points, consider them to be on the top or bottom edges.

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

```
Enter rectangle bounds: 2 15 4 20
Enter point coordinates: 0 6
Point is outside
```

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

```
Enter rectangle bounds: 2 15 4 20
Enter point coordinates: 6 8
Point is inside
```

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

```
Enter rectangle bounds: 2 15 4 20
Enter point coordinates: 2 12
Point is left edge
```

Important: Do *not* change any of the code in the `main` function.

Hint: Break the problem down into multiple `if`/`else if` cases to check each of the conditions.