CS 101, Spring 2013 — Mar 26th — Exam 2 Name: ____________________

Question 1. [2 points] Given the following array, what is the value of `quiz[5]`?

```
float quiz[5] = {88.9, 56.5, 92.1, 78.0, 68.9};
```

A. Can't be predicted

B. 0

C. 68.9

D. 88.9

Question 2. [3 points] Given the following code snippet, what is the expected output of the `printf` statement?

```
int deca[3] = {12, 2, 0};
deca[2] = deca[0] + deca[1];
printf("%i", deca[2]);
```

A. Can't be predicted

B. 0

C. 12

D. 14

Question 3. [3 points] From the choices below, circle *all* valid function prototypes (there may be more than one).

A. `int sumNum(int x, y, z);`

B. `float avg(float exam1, int exam2);`

C. `void drawSq(int squareHeight);`

D. `printLine(size);`

Question 4. [6 points] Add code to the following program just below the `TODO` comment so that the loop containing the `printf` statement will print the output `17 42 121`. Use assignment statments to set the values, i.e. do **not** use `scanf` to obtain the inputs.

```
#include <stdio.h>

int main(void) {
  int arr[10];
  int m, n;

  // TODO









  for (int i = m; i <= n; i++) {
    printf("%i ", arr[i]);
  }
  printf("\n");

  return 0;
}
```

Question 5. [6 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

int findMin(int a, int b);

int main(void) {
  int x = 6, y = 7;
  int min = 0;
  findMin(x, y);
  printf("%i\n", min);
  return 0;
}
```

```
int findMin(int a, int b) {
  int min;
  if (a < b) {
    min = a;
  } else {
    min = b;
  }
  return min;
}
```

Question 6. [8 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
#include <stdio.h>

#define N 6

int multiply(int x, int y) {
  return x * y;
}
```

```
int main(void) {
  int a[N] = {2, 4, 6, 8, 10, 12};
  int product = 0;
  printf("Start Here!\n");

  for (int i = 0; i < N; i++) {
    product = multiply(a[i], i+1);
    printf("Multiply %i: %i\n",
           i, product);
  }

  printf("All Done!\n");
  return 0;
}
```

For Questions 7–12, circle True or False.

Question 7. [2 points] True or False: A function can use a `return` statement to return more than one value.

Question 8. [2 points] True or False: All functions must return a value.

Question 9. [2 points] True or False: A function call may have more argument expressions than there are function parameter variables.

Question 10. [2 points] True or False: Variables defined in the body of the `main` function can be accessed by any other function in the program.

Question 11. [2 points] If a function with a non-`void` return type does not execute a `return` statement, then the function will automatially return the value `0`.

Question 12. [2 points] It is legal to declare a variable with the same name in the bodies of two different functions.

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++** 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_Exam2
```

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

Editing code: Use Notepad++ to open the source file (e.g., `question13.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 13, run the following command in Cygwin Terminal:

```
make question13.exe
```

Change the number as appropriate for the other questions (e.g., `question14.exe`).

Running: To run the program for Question 13, run the following command in Cygwin Terminal:

```
./question13.exe
```

Change the number as appropriate for the other questions (e.g., `question14.exe`).

To submit: In Cygwin Terminal, run the command

```
make submit
```

Enter your Marmoset username and password when prompted.

Good luck!

Question 13. [15 points] The program `question13.cpp` reads a single `int` value into a variable called `num_values`. It then reads that many `double` data values into an array called `values`. After that, it reads another `int` value into a variable called `op`, which represents an operation to be performed on the values in the `values` array.

Your task is to modify the program so that its output is a line of text of the form

`Result is` N

where N is a value computed as follows:

- If `op` is 1, N should be the sum of the elements of the `values` array
- If `op` is 2, N should be the product of the elements of the `values` array
- If `op` is 3, N should be the maximum of the elements of the `values` array
- If `op` is any other value, N should be -1

Example inputs/outputs:

Input			Expected output
2	2.6 8.2	1	`Result is 10.800000`
4	2.6 8.2 5.3 7.1	1	`Result is 23.200000`
2	2.6 8.2	2	`Result is 21.320000`
3	2.6 8.2 5.3	2	`Result is 112.996000`
5	2.6 8.2 5.3 7.1 2.2	3	`Result is 8.200000`
2	2.6 8.2	4	`Result is -1.000000`

Hints:

- The code to read the input values is provided for you
- You do *not* need to use a function for this program
- Use an `if`/`else if`/.../`else` statement to check the operation value and decide which operation to perform on the array values
- Each operation will need to use a loop to process the values in the array

Make sure the output is in *exactly* the format described above.

Question 14. [15 points] Modify the program `question14.cpp` so that when the user enters an integer n, the program prints an n by n block of asterisk characters. For example, if the user enters the integer 5, then the output of the program should be

```
*****
*****
*****
*****
*****
```

Requirement: the only change you may make to the `main` function is to uncomment the line which reads

```
//draw_square(n);
```

so that it reads

```
draw_square(n);
```

If you make any other changes to the `main` function, you will not receive any credit!

Hint: add a function prototype and a definition for the `draw_square` function.

Question 15. [15 points] A "spaceship" figure can be created from text characters as follows:

```
<<<--O-->>>
```

A spaceship consists of m less-than (`<`) characters, followed by n hyphen (`-`) characters, followed by an `O` (a capital O, not the number 0), followed by n hyphens, followed by m greater-than (`>`) characters. The figure above is the case where $m = 3$ and $n = 2$. A different spaceship, where $m = 4$ and $n = 6$ would look like this:

```
<<<<------O------>>>>
```

Modify the program `question15.cpp` so that it prompts the user for the values of m and n and then draws the appropriate spaceship.

Important: Your program **must** draw the spaceship by making calls to the `print_chars` function included in `question15.cpp`. You must modify the code in the `main` function, but **do not** modify the code in the `print_chars` function.

Hint: a literal character can be specified using single quotes. For example,

`'<'`, `'>'`, `'-'`, and `'O'`

are the literal `<`, `>`, `-`, and `O` characters respectively.

Question 16. [15 points] In the file `question16.cpp`, complete the program as follows to implement a simple "grass fire" simulation. The program should read exactly 10 integer values, each of which will be `0` or `1`. The `0` values represent grass, and the `1` values represent grass that is on fire.

The rules of the simulation are as follows: at each time step, a location containing grass will catch on fire if either of its neighbors (left or right) are on fire. Locations that are on fire will continue to burn.

The output should be a series of lines consisting of `,` (comma) characters representing grass and `*` (asterisk) characters representing fire. The first line of output represents the original input. Each subsequent line represents a subsequent generation. There should be exactly 5 lines of output, for the original configuration and the 4 subsequent generations.

For example, if the input to the program is

```
1 0 0 0 0 1 0 0 0 0
```

then the output should be

```
*,,,,*,,,,
**,,***,,,
********,,
*********,
**********
```

Hints/specifications:

- The program should consider both the first location's left neighbor and the last location's right neighbor to be grass (not on fire)
- You will probably want to use two arrays, one to represent the current generation and one to represent the next
- Make sure your output *exactly* matches the format shown; each line should consist of only `,` and `*` characters