CMSC 216 Quiz 2 Worksheet

The next quiz for the course will be on Wed, Sep 17. The following list provides additional information about the quiz:

- Do not post any solutions to this worksheet in Piazza. That represents an academic integrity violation.
- The quiz will be a written quiz (no computer).
- The quiz will be in lab session.
- Closed book, closed notes quiz.
- Answers must be neat and legible.
- Quiz instructions can be found at http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html
- Make sure you know your section number and your TA's name.

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TA or instructor during office hours. If is recommended that you try this exercises on paper first (without using the computer).

Exercises

1. What is static storage? Which kind of variables have this kind of storage?
2. What is linkage? Which C keyword allow us to change the linkage property of a function?
3. Draw a map illustrating the different memory areas associated with a C program. This is the diagram we draw in class.
4. Are only Object-Oriented languages allowed to have a heap?
5. Can C functions be overloaded?
6. What is a prototype?
7. How do you run the debugger (gdb)?
8. How can you detect where in a program a segmentation fault is taking place by using the gdb debugger?
9. How can you print the stack using gdb?
10. Which flag we need to use in order to compile code that can then be used with the debugger?
11. Which gdb command do you need in order to set a breakpoint at a function named do_task?
12. Which gdb command do you use to execute a statement?
13. Which gdb command do you use to print the value of a variable?
14. Which gdb command do you use to list the code associated with a function named do_task?
15. Which gdb command do you use to run a program so the input is from a file called public01.in instead of standard input?
16. Write a C function that determines whether a positive sequence of integer values provided by the user represent an increasing sequence. For example, 3, 6, 10 represents an increasing sequence. The function will return true if the sequence is increasing and false otherwise. You can assume a negative value will mark the end of the sequence. You may not use arrays for this problem and your function must work for any number of values (not just 3).

17. Define a function named **read_and_compute_prod** that has the prototype below. For this problem:

    - The function computes and returns the product of integer values provided by the user.
    - Use scanf to read the values.
    - You don't need to display any prompt or message as each value is read.
    - The program will stop reading values when the value provided by the user is 0, or if it corresponds to the parameter value (stop). Notice the parameter does not represent the number of values to read; it represents when to stop.
    - The stop value is not part of the product.

    For example, calling **read_and_compute_prod(-1)** will return 54 if we enter the values **2 3 9 0** or the values **2 3 9 -1**

    int read_and_compute_prod(int stop)

18. Define a function named **sum_of_divisibles_by** that has the prototype below. For this problem:

    - The function reads two integer values (lower limit and upper limit) that represent a range. For this problem you can assume the first value will always be lower than or equal to the second.
    - The function computes the sum of values in the range that are divisible by the specified parameter value. Notice that the range includes the lower and upper limit values.
    - The function will display the message "Enter lower and upper limit: " while reading the range values.
    - The following driver and associated output illustrates the functionality expected from the function you need to write. Keep in mind this is just an example (your function must work for different sets of values and not just the ones presented in the example.) In the example, underlined text is input the user provides and % is the Unix prompt.

Driver

```
int main() {
   printf("Sum of divisible by 2: %d\n", sum_of_divisibles_by(2));
   printf("Sum of divisible by 3: %d\n", sum_of_divisibles_by(3));

   return 0;
}
```

Output

```
% a.out
Enter lower and upper limit: 2 7
Sum of divisible by 2: 12
Enter lower and upper limit: 2 9
Sum of divisible by 3: 18
%
```

```
int sum_of_divisibles_by(int divisible_by)
```