CMSC 216 Quiz 1 Worksheet

The first quiz for the course will be on Wed, Feb 7. The following list provides additional information about the quiz:

- 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.
- You must use a pencil.
- Quiz instructions can be found at http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html
- **Regarding Piazza -** Feel free to post questions in Piazza regarding the worksheet and possible solutions to problems.

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 TAs or instructors during office hours. It is recommended that you try these exercises on paper first (without using a computer).

Exercises

1. Make sure you have read the information available at:

   http://www.cs.umd.edu/class/resources/academicIntegrity.html

2. When is a project considered late in this class?

3. Write a Unix command that will take you to your home directory.

4. Write a Unix command that will copy a directory called **data** present in your home directory to the current directory. You can assume the current directory is different than the home directory.

5. Write a Unix command that will copy the folder Week1 present in the lecture_examples folder of the ~/216public directory to your ~/216 directory of your home directory.

6. Write a Unix command that will copy the folder **commands_info** present in the **lecture_examples** folder of the ~/216public directory to your ~/216 directory of your home directory. The command should work when executed from any directory (e.g., you cannot assume your current directory is ~/216).

7. What is the size (in bytes) of a char type?

8. What is the output of the following program?

```
#include <stdio.h>

int main() {
   int x;

   printf("%d\n", x);

   return 0;
}
```

9. Suppose you write a C program and it has an infinite loop; how do you stop the program?

10. What possible problem(s) are associated with the following code fragment?

```
int x;
scanf("%d", x);
```

11. For this quiz, you will need to provide examples of academic integrity violations. The following is the list you need to know:
    a. Hardcoding of results in a project assignment. Hardcoding refers to attempting to make a program appear as if it works correctly (e.g., printing expected results for a test).
    b. Using any code available on the internet/web or any other source.
    c. Hiring any online service to complete an assignment for you.
    d. Sharing your code or your student tests with any student.
    e. Using online forums (other than Piazza) in order to ask for help regarding our assignments.

12. Does the following code compile? Briefly explain.

```
#include <stdio.h>

void p1(int x) {
    printf("%d", x);
}

void p1(float x) {
    printf("%d", x);
}

int main() {
   p1(2);
   return 0;
}
```

13. 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 once 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);
```

14. Define a function called **print_table** that has the prototype below. For this problem:

- The function reads two integer values (using scanf) and prints the square of each value in the range. For each value in the range the function will print the value, followed by the word **even** if the value is **even** and **odd** otherwise, followed by the square.
- Use the prompt "Enter range values: " to read the two integer values.
- If the print_header parameter is true the function will print the header "Squares Table" before any value in the range is processed.
- You can assume the first integer value is less than or equal to the second one.
- **The function will return the number of values in the range.**

```
int print_table(int print_header);
```

The following is an example of calling the function you are expected to write. The user entered the values 4 and 10 (values that are underlined). The % represents the Unix prompt.

Driver

```
int main() {
   print_table(5);

   return 0;
}
```

Output

```
% a.out
Enter range values: 4 10
Squares Table
4 even 16
5 odd 25
6 even 36
7 odd 49
8 even 64
9 odd 81
10 even 100
%
```

15. Define a function called **print_powers** (int print_powers(int limit)). For this problem:

- The function will read a character (either **f** or **i**). If the user enters **f**, the function will print the powers of numbers from 1 up to the **limit** value (specified in the parameter) in increments of **.5**. If the user enters **i**, the function will print the powers of values from 1 up to the **limit** value in increments of two.
- Use scanf to read the character. The function will display the following message in order to read the character: **"Enter f (float) or i (integer): "**
- You can assume users will enter correct data (either **f** or **i**).
- Notice the output format is important. See the example we have provided below. If the user enters i, data is displayed as integer values; otherwise as float values.
- You can assume the **limit** parameter will be greater than 1. Notice that the output might not include the limit value (e.g., user enters **i**, but the limit value is 6).
- **The function will return how many powers were printed.**
- 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. Notice we are running the program twice and we are not using the value returned by the function. For the first program execution, the function returns 9; for the second the function returns 3.

Driver

```
int main() {
   print_powers(5);

   return 0;
}
```

Output

```
% a.out
Enter f (float) or i (integer): f
1.000000, 1.000000
1.500000, 2.250000
2.000000, 4.000000
2.500000, 6.250000
3.000000, 9.000000
3.500000, 12.250000
4.000000, 16.000000
4.500000, 20.250000
5.000000, 25.000000
% a.out
Enter f (float) or i (integer): i
1, 1
3, 9
5, 25
%
```

16. Define a function called **compute_avg** (prototype can be found on the next page). For this problem:

- The function will read an integer representing a number of scores to process.
- The function will read the specified number of scores (float values), compute the average, and return the average.
- The message "Number of scores:" should be used when reading the number of scores.
- The message "Score:" should be used when reading each score.
- If the **display** parameter is true the function will also print the following message:

  Average for <number_of_scores> scores: <average>

  where <number_of_scores>and <average> corresponds to the number of scores and the computed average, respectively.

- The average must be printed using two digits after the period.
- Do not provide a main() function; you do not need to add any #include preprocessor directives.

- 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("%f\n", compute_avg(1));

   return 0;
}
```

Output

```
% a.out
Number of scores: 3
Score: 75
Score: 80
Score: 90
Average for 3 scores: 81.67
81.666664

float compute_avg(int display)
```