CMSC 216 Quiz 1 Worksheet

The first quiz for the course will be on Wed, Jun 5. The following list provides additional information about the quiz.

- The quiz will be a written quiz (no computer).
- Closed book, closed notes quiz.
- The quiz will be in lab session.
- Answers must be neat and legible.
- 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.
- You must take your quiz in your assigned lab/discussion session and not show up to a random discussion session. We will not grade quizzes taken in the incorrect session.
- We use the Gradescope system to grade your quizzes after they have been scanned. For the system to recognize your work, you need to print your name (uppercase) and student id. The following is an example of the information you need to provide in your quiz:

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE): Mary, Smith

STUDENT ID (e.g. 123456789): 123456789

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 instructor during office hours. It is recommended that you try these exercises on paper first (without using the computer).

Exercises

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

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

2. 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.

3. Write a Unix command that will list of the files present in the current directory that end with the .c extension.

4. Write a single Unix command that will copy to the 216 directory (present in your home directory), the files that end with the .txt extension that exist in the parent directory of the current working directory.

5. Write a Unix command (in your ~/216 directory) that will copy the file ex.txt present in the ~/tmp directory to:
   a. The current directory.
   b. Your home directory.
   c. To the parent's directory of the current directory.
   d. To the root directory (you can assume you can copy to the root directory).

6. Write a Unix command that will copy the folder commands_info present in the lecture_examples folder of the ~/216/public 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. Always lock your computer when you leave it alone (e.g., going to the restroom in lecture), otherwise bad things could happen. 😊 Do you realize that if you leave your computer open, anyone can execute submit in your project directory and steal your code?
9. 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**

```c
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
%
```

10. Define a function called `compute` (prototype below) that computes either the sum or product of integers provided by the user. For this problem:

- The function reads integers values using scanf and the message "Enter value: " to read each value.
- The function will stop reading values when the user provides -1.
- If the sum_flag parameter is true, the function will compute and return the sum of the values; otherwise it will compute and return the product.
- If the print_flag parameter is true, the function will print the computed valued before returning the value. Use the message "Result: " followed by the computed valued.
- You can assume users will provide valid data (integers) and the computation will not cause an overflow.
- Below we provided an example of using the function. Underlined text represents input and % the Unix prompt.

**Driver**

```c
int main() {
    compute(0, 1);
    return 0;
}
```

**Output**

```
% a.out
Enter value: 4
Enter value: 5
Enter value: -1
Result: 20
%
```

```c
int compute(int sum_flag, int print_flag)
```
11. Define a function called **phone_password** (prototype below) that reads a phone number and verifies it corresponds to the phone number specified in the parameters. We define a phone number as a three digit integer, followed by a dash, followed by a four digits integer. For this problem:

   a. The function reads two integer values separated by a dash. The first integer represents the three digits of a phone number and the second the four digits; we are not using area code.
   b. The message "Enter phone in XXX-YYYY format:" will be displayed before reading the values.
   c. The function will compare the three digits and four digits values provided by the user against the three_digits and four_digits parameters. The function will keep asking for a phone as long as the values provided do not correspond to the parameters. Once they are the same, the function will end returning the number of attempts. Notice that the expected value represents one attempt.
   d. If the error_message parameter is true, the function will print the message “Invalid phone” after an invalid phone has been provided.
   e. You can assume users will provide valid data (integers) and a dash in between the integers.

   int phone_password(int three_digits, int four_digits, int error_message)

12. Complete the **get_taxes** function below. The function reads a character (F or H) indicating whether someone is a full time or half time employee, and a float representing the employee's salary. The values are separated by a dash (e.g., F-50000). The function will return the amount of taxes the employee will pay based on the following rules: a half time employee will pay 0 taxes; a full time employee will pay 20% of the specified salary. If the print_values parameter is true, the function will print (after the message “Values provided:”) the values read before returning the tax amount. If the user provides invalid input (e.g., - is missing, or a word is provided instead of a number) the function will print the message "Invalid input" and return -1.0. You must use scanf to read all values. The following are examples of calling the function using the provided main.

   int main() { printf("%f\n", get_taxes(1)); return 0; }

   % a.out
   F-50000
   Values provided: F, 50000.000000
   10000.000000
   %

   % a.out
   H-50000
   Values provided: H, 50000.000000
   0.000000
   %

   % a.out
   F 50000
   Invalid input
   -1.000000
   %

   float get_taxes(int print_values)