R2: C Language Basics

General Instructions:

1- Create a C project in eclipse called R2
2- Extract the file R2.zip into a folder of your choice at your machine.
3- Copy all files (other than this PDF file) into the R2 project.
4- Rename the file: "R2_template.txt" to "R2.c".
5- Enter your credentials on top of the file "R2.c".
6- Follow the videos posted by the instructor on how to complete the tasks presented in this worksheet.
7- To test your code, select the proper function in the file: "R2_test.c".
8- Compare your results with those presented in "R2_output.txt", using the website https://text-compare.com/ (Note: the videos use a validator to compare the results, use the above website instead).
9- When you are done, you need to submit ONLY your "R2.c" file. Do not export the project or upload a .zip file.

Task 1: Factorial

Implement the function unsinged long `factorial(int)` which returns the factorial of a given integer. The function needs to be implemented using recursion. If the given input is less than 0, the function prints the following error msg, and returns 0:

```
Error(factorial): Invalid input
```

Task 2: Solving a mathematical Equation:

Write a function, called solve, that takes four input parameters: r, y, x and b. The function computes the result, stored in variable a, according to the following equation.

$$a = \left\lfloor \frac{r^3 - y}{x} - \sqrt{b^4} \right\rfloor$$

Use the built-in functions defined in the library `math.h` to compute the powers and square root. Watch out for precedence rules and the data types returned by the mathematical functions.

The function prints the output in a format similar to the following:

```
[r = 2, y = 3, x = 2, b = 2] --> a = -2.0
```

The function also returns the value of a, which is of type double

Task 3: Detecting Non Whole Numbers

Implement the function `int is_double(double)` which checks if a given number is a whole number or not. If the number is whole, e.g. 1.0 or 200.00, the function returns 0. If the number is not whole (double), the function returns 1. Note that double here does not mean the data type double, it simply means non-whole numbers.

Task 4: Detecting a fraction:

Implement the function `int is_fraction(double)` which checks if a given number is a fraction in the range (-1,1), excluding 0. Examples of fractions include: 0.34, -0.69, 0.9999. Examples of invalid fractions include -1, 1 ,2.3 ,1.4 and 0.

If the given number is a fraction the function returns true. Otherwise, returns false

Task 5: Find Number of Days in a Month:

Implement the function `void get_month()` which finds the number of days in a given month.

The function asks the user to enter a number, and the uses switch statement to print one of the following outputs:

- 31 days
- 30 days
- 28/29 days
- Invalid input

The function receives no parameters and makes no returns.

Task 6: Classify Number 1:

Implement the function `void classify_num1(int)` which classifies an input integer in terms of being even/odd and positive/negative. Implement the function using the ternary operator.

The function prints one of the following outputs:

- This is an even positive number
- This is an even negative number
- This is an odd positive number
- This is an odd negative number

The function does not provide a proper classification for the number 0. The function does not return any value.

Task 7: Classify Number 2:

Implement the function `void classify_num2(int)` which classifies an input integer in terms of being even/odd and positive/negative. Implement the function using the *`goto`* command. The function prints one of the following outputs:

- This is an even positive number
- This is an even negative number
- This is an odd positive number
- This is an odd negative number
- This is a zero..

The function does not return any value.

Task 8: Odd-Even Counter

Implement the function `count_odds_evens` which is provided to you by the instructor.