CMSC 223 Systems Programming - Lab 2

C data types and limits. Computing with dates.

1. Write a C program to figure out how many bytes are used by the following types on your computer/compiler (Hint: use the sizeof() operator:

   - short int
   - int
   - long int
   - long long int
   - char
   - float
   - double
   - long double

Confirm your answer by looking at the contents of file: /usr/include/limits.h

2. You have learned that to store a signed integer, computers use 2’s complement representation. Thus, in 2’s complement representation, the largest positive integer that can be stored in \( x \) bits is the value \( 2^{x-1} - 1 \) and the smallest value is \( -2^{x-1} \). For example, when 16 bits (2 bytes) are used, the largest value that can be represented is \( 2^{15} - 1 \), and \( -2^{15} \), giving us the range \([-32,768..32,767]\).

   Armed with the above knowledge, let us consider the loop below:

   ```c
   int start = 0, end = 10;
   for (int i = start; i < end; i++)
       printf("i = %d\n", i);
   ```

   The loop above, when executed will print out values of \( i \) in the range \([0..9]\). Go ahead, write a program, or add the above to the program from (1) above and run it.

3. Carefully, examine what happens in the loop above when the value of \( i \) is 9. Since 9 is less than \( end \) (10), the condition is true and therefore it will be printed. After that, \( i \) is incremented to 10. This time, \( i \) is no longer less than \( end \), so the loop stops or terminates.

   So far, so good. Next, let us change the loop above to push to the limit of the values int variables can take. For (1) you determined the number of bits an int variable takes. With that, and information in (2) above, you can calculate the range of values the variable \( i \) will take. Go ahead and write it down below (Hint: see you text if needed):

   range of values for int variables:__________________________
Let’s assume the range is denoted by \([\text{LOW}..\text{HIGH}]\). Rewrite the loop as shown below:

```c
int start = HIGH-5, end = HIGH;
for (int i = start; i < end; i++)
    printf(“i = %d
”, i);
```

Fill in the value of \(\text{HIGH}\) that you obtained above. Run the program.
Describe the output...

Next, change the loop as shown below (we have changed the < to \(\leq\)):

```c
for (int i = start; i <= end; i++)
```

Run the program again. What happens?
[Hint: You may need to stop the program by typing CTRL-c !]

Make sure you understand what happens and why.

4. Write a C program that given a date, computes what day of the year the date corresponds to. For example, it behaves as follows:

Enter a date: 1/29/2023
The date 1/29/2018 is the 29th day of the year.

Enter a date: 9/11/2023
The date 9/11/2023 is the 254th day of the year.

You will need to determine, given a year whether it is/not a leap year. You may use the function below:

```c
int leapYear(int year) {
    return ((year%4==0) && (year%100!=0)) || (year%400==0));
} // leapYear()
```

Huh? Please read the definition of a leap year coded above carefully and, if needed, consult an online source on what is a leap year. In your program, this \text{leapYear()} function should appear before the definition of the \text{main()} function.

If you have time, can you fix the output so that they are printed as shown below:

Enter a date: 1/29/2023
The date 1/29/2023 is the 29th day of the year.

Enter a date: 9/11/2023
The date 9/11/2023 is the 254th day of the year.

That is, proper suffixes are used (st, nd, rd, th).