Lab 6: More Functions and Variables

1. Finding the perfect number

Let's check if a number is "perfect". In math, a perfect number is one which equals the sum of its "proper" factors. For example, 6 has proper factors 1, 2, and 3; and as it happens 6 = 1 + 2 + 3 so 6 is a perfect number. Another example is 12 with proper factors 1, 2, 3, 4, 6 but since $12 \neq 1 + 2 + 3 + 4 + 6$, it means 12 is not a perfect number.

For this activity, you are encourage to work with another student. You can help each other write the code, and figure out how to make it work. **Plus, coding together is more fun!** If you are working online, post a message into the Discord general channel: “LFG Lab 6!” (optional).

1. Overview of what we are going to do (detailed step-by-step instructions are below):
   - First we'll create a function to test for proper factors:
     `bool isProperFactor(int num, int divisor)`
   - Then we'll sum up proper factors:
     `int sumProperFactors(int n)`
   - Finally, we'll test if a number is perfect:
     `bool isPerfectNumber(int n)`
   - These functions will call one another as needed!
2. Create a new file named `lab6.cpp`.
3. Crate the following function:
   `bool isProperFactor(int num, int divisor);`
   - Make this function return `true` if both of the following are true:
     - `divisor` < `num`, and
     - The `divisor` divides evenly into `num`:

       Hint: use % to check for no remainder .
   - otherwise, return `false`.

     Math theory: *d* is a proper factor of *n* if *d* < *n*, and *d* is a factor of *n* (which means, *d* divides evenly into *n*).
   - *Style hint*: Instead of writing code that uses an `if`-statement to return true or return false, you should return a boolean expression (which will evaluate to true or false):

```
// Don't do it this way
bool isEven(int n)
{
    if (n % 2 == 0) {
        return true;
    } else {
        return false;
    }
}
```

```
// Do it this way!
bool isEven(int n)
{
    return n % 2 == 0;
}
```

The code on the left has more logic to write, read, and debug.

4. Write a couple test statements in `main()` to test `isProperFactor()`
   - First output to the screen if 3 is a proper factor of 6:
     ```
     cout << "3 proper factor of 6? " << isProperFactor(6, 3) << endl;
     ```
   - `perfect.cpp` Repeat the test for the following pairs:
     - 4 a proper factor of 6?
       1 a proper factor of 6?
       6 a proper factor of 6?
   - Your output should look similar to this:
     ```
     3 a proper factor of 6? 1
     4 a proper factor of 6? 0
     1 a proper factor of 6? 1
     6 a proper factor of 6? 0
     ```
5. **Exam Note**
   You will likely be asked to write functions on exams. For example, if you were asked to implement the `isProperFactor()` function on an exam, the question might be:

   > Implement the `isProperFactor()` function which accepts two integers (`int`) in the order: 1) the number-to-test, and 2) the divisor. It returns `true` if both of the following are true:
   >
   > - divisor < number-to-test, and
   > - the divisor divides evenly into the number-to-test.
   >
   > Otherwise return `false`.
   >
   > For example:
   >
   > `isProperFactor(30, 5)` → returns `true`
   > `isProperFactor(30, 7)` → returns `true`
   > `isProperFactor( 8, 8)` → returns `false`

   - Note that you are asked to *just* implement the `isProperFactors()` function:
     - You would *not* do any input from the keyboard or output to the screen: all values are "accepted" as parameters.
     - You would *not* write the code for the `main()` function. If you need to write `main()` you will be explicitly asked to do so.
     - Depending on how the question is worded, you may or may not have to write any necessary `#include` statements (in this case, none are needed).
     - *Likely don't* need to write comments during the exam unless told to do so.
     - As always, good variable and function names matter; no magic numbers.

6. Create a function to sum up the proper factors of a number:
   `int sumProperFactors(int n)`
   - Return the sum of all numbers between 1 and `n` which are proper factors of `n`.
   - **On paper:** The exam involves writing code on paper; try writing this function on paper first. It's a small function, but good practice.
   - **On the computer**: Now write it on the computer. Compare to your on-paper solution.
   - Hints:
     - Create a local variable to hold the sum of the values. Try naming it `sum`.
     - Create the loop. Try a `for` loop because you know where it starts and stops.
     - In the loop, call the `isProperFactor()` function to check each number if it is a proper factor. If the number is a proper factor, add it to the sum.
     - **Return** the sum from the function.
7. In `main()`, prompt the user and read in an integer. This value will be used to see if it's a perfect number.
   - *Hint*: Name the variable something like `checkNum`.
8. In `main()`, print the sum of proper factors of `checkNum` by doing the following:
   - In `main()`, call `sumProperFactors()` passing in `checkNum`
   - In `main()`, display the sum to the screen. Complete output should now look like:

```
3 proper factor of 6? 1
4 proper factor of 6? 0
1 proper factor of 6? 1
6 proper factor of 6? 0
Enter a number to check: 15
Sum of proper factors: 9
```

9. Finally, create a function with the following header:
   `bool isPerfectNumber(int n)`
   - Make this function return `true` if `n` is a perfect number (i.e. where the proper factors of *n* add up to equals *n*; otherwise return `false`).
     - *Challenge:* How short and concise can you make this function's implementation? Don't make it cryptic, just make it concise!
   - Call `isPerfectNumber()` from `main()`, passing in the value the user entered. In `main()`, print a message stating if it is, or is not, a perfect number. Here are two sample outputs:

```
3 proper factor of 6? 1
4 proper factor of 6? 0
1 proper factor of 6? 1
6 proper factor of 6? 0
Enter a number to check: 15
Sum of proper factors: 9
I'm sorry, that's not a perfect number.
```

```
3 proper factor of 6? 1
4 proper factor of 6? 0
1 proper factor of 6? 1
6 proper factor of 6? 0
Enter a number to check: 6
Sum of proper factors: 6
Amazing! It's perfect!
```

10. Using your program, find a perfect number between 490 and 500.
    - You might start by typing in the numbers to see if you can find it.
        - Is there a better way you could find perfect numbers?
    - Modify your program to output every perfect number between 1 and 10,000. (You should find 4, slowly...).
11. *Optional Challenge*: How long does it take your program to find the 5th perfect number? *Hint*: Use `INT_MAX` from `#include <climits>` as the upper bound for where to stop.
    - You may want to have your program output something to the screen after every 100,000 numbers checked, just so you know it's alive.
    - Extra challenge: Make your program display its status every 10 seconds to see what number it's checked. Hint: use `time(0)`, which gives the time in seconds.
    - While it's running, look up online what number is the 5th perfect number. Guess how long it will be before your computer finds the answer.
        - Let your program run for a while; does it seem to speed up or slow down? Explain why it does this.
        - This highlights the need for efficient algorithms. Fast computers are not the answer: we need to compute smarter, not faster. It can make a HUGE difference.
        - For fun, compute how old you'll be when your computer finds the 6th, 7th, or 8th perfect number! Change your program to output how many numbers it checked in the last 10 seconds (count each number checked, every 10 seconds print the count and reset the count).
    - How can you speed up the algorithms used here? Can you think of any optimizations?

12. Understanding
- Describe (in words) what your main function would look like if you implemented it without using functions. What is the advantage of using functions?
- What would happen if the function `isProperFactor()` *displayed* the answer instead of *returned* the answer? Would your program work?
- Inside `sumProperFactor(),` why is a `for` loop a good type of loop to use?
- Why are the following two lines of code interchangeable? Which is better? Why?

```
if (isProperFactor(n, i)) {...}
if (isProperFactor(n, i) == true) {...}
```

2. Lab credit

Submit the following to CourSys to get credit for the lab:

- Complete above steps in one file. OK to skip any “optional” sections.
- Comment your code correctly: comment for every ~2-5 lines of code.