CS 2100: Data Structures
Homework 1

Due *via git* by 11:59pm on Tuesday, Jan. 23

This is a homework you must complete individually. Each problem here is focused on learning the basic syntax of C++. You should submit a separate cpp file for each problem, commented with your name and the problem number at the top (as well as other comments throughout which describe the program as necessary). Note that these should be tested before submitting them; make sure they actually compile and work!

Please type all answers and submit via git by the date and time specified.

1. Write a short C++ program that presents the user with a choice of your 5 favorite beverages (tea, coffee, Coke, water, or whatever). Then prompt the user to choose a beverage by entering a number 1-5. Output which beverage they chose. If they enter a number other than 1-5, output "Error: Choice was not valid."

   Note: You can assume that the user will enter a value integer, so you don't have to deal with error handling if they enter something totally crazy.

2. Write a function `printSquare` which takes as input an integer `size` and a character `fillChar`, and prints a square of the specified size formed from that character. For example, if `size` is 4 and `fillChar` is @, then it should print:

   ```
   @@@@
   @@@@
   @@@@
   @@@@
   ```

   Write a (short) main which calls this function at least twice with different inputs to verify that it works correctly.

3. The following exercise is designed to get you used to pre and post increment, as well as the (somewhat lazy) way that C++ handles some of the basic types.

   Write a program that shows the result pre and post increment and decrement (so the ++val and val++, and –val and val–). However, try these operations on a character, and not a number, to demonstrate how char and int are not distinct in this language. Your program should prompt for a character to be entered, and then print the results.

   For example, a sample run of your program (after compiling) might look like:

```
echambe5@hopper:~/cs2100/spring18$ ./a.out
Enter a letter: n

Display the operation of pre and post increment and decrement :
------------------------------------------------------------------------
The letter is : n
If I use  post increment (letter++) the number is now: n
And now my letter is: o
Now if I pre-increment (++letter) in my cout I get p
 And now my letter is: p
If I use  post decrement (letter--) the number is now: p
 And now my letter is: o
Now if I pre-decrement (--letter) in my cout I get n
 And now my letter is: n
```

4. A number n is *perfect* if it is equal to the sum of its proper divisors, where a proper divisor is a number which divides n and is less than n. For example, 6 is a perfect number because $1 + 2 + 3 = 6$, and 1, 2, and 3 are the only divisors less than 6. Similarly, 28 is perfect also, because $1 + 2 + 4 + 7 + 14 = 28$.

   Write a function isPerfect that takes an input n as input and returns the boolean true if n is perfect, and false otherwise. Then test your function by writing a main program that checks for all perfect numbers in the range 1 to 9999 by testing each number one at a time. When a number is found to be perfect, your program should print it to the screen.

   Note: The first two lines of output should be 6 and 28, and your program should find two other perfect numbers as well.