NYU, Tandon School of Engineering
Extended Bridge to CS — Summer 2021

Homework #6
Due by Friday 8/20 11:59pm

Submission instructions:

1. For this assignment, you should turn in 5 files:
   - Four '.cpp' files, one for each question 1-4.
     Name your files 'YourNetID_hw6_q1.cpp', and 'YourNetID_hw6_q2.cpp', etc.
   - One '.pdf' file with your answers for question 5.
     Name your file 'YourNetID_hw6_q5.pdf'

2. You must type all your solutions. We will take off points for submissions that are handwritten.

3. **You should submit your homework in the Gradescope system.**

4. **You can work and submit in groups of up to 4 people. If submitting as a group, make sure to associate all group members to the submission on gradescope.**

5. Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose the most appropriate control flow statements, break down your solutions by defining functions, etc.

Question 1:

The Fibonacci numbers sequence, F_n, is defined as follows:
F_1 is 1, F_2 is 1, and $F_n = F_{n-1} + F_{n-2}$ for n = 3, 4, 5, ...
In other words, each number is the sum of the previous two numbers. The first 10 numbers in Fibonacci sequence are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
Note: Background of Fibonacci sequence: https://en.wikipedia.org/wiki/Fibonacci_number

1. Write a function `int fib(int n)` that returns the *n-th* element of the Fibonacci sequence.
2. Write a program that prompts the user to enter a positive integer `num`, and then prints the `num`'s elements in the Fibonacci sequence.
   Your program should interact with the user **exactly** as it shows in the following example:
   Please enter a positive integer: 7
   13

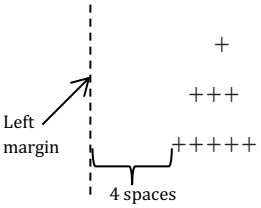
Question 2:

Write a program that, prints a 'pine tree' consisting of triangles of increasing sizes, filled with a character (eg. '*' or '+' or '$' etc).
Your program should interact with the user to read the number of triangles in the tree and the character filling the tree.

Your implementation should include the following functions:

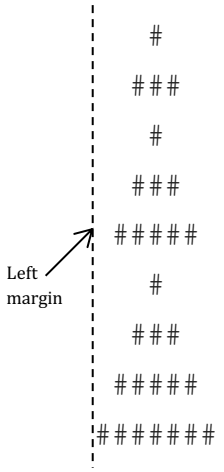
a. `void printShiftedTriangle(int n, int m, char symbol)`
   It prints an `n`-line triangle, filled with `symbol` characters, shifted `m` spaces from the left margin.

   For example, if we call `printShiftedTriangle(3, 4, `+`)`, the expected output is:



b. `void printPineTree(int n, char symbol)`
   It prints a sequence of `n` triangles of increasing sizes (the smallest triangle is a 2-line triangle), which form the shape of a pine tree. The triangles are filled with the `symbol` character.

   For example, if we call `printPineTree(3, `#`)`, the expected output is:



Question 3:

The number ***e*** is an important mathematical constant that is the base of the natural logarithm. ***e*** also arises in the study of compound interest, and in many other applications.

Background of e: https://en.wikipedia.org/wiki/E_(mathematical_constant)

e can be calculated as the sum of the infinite series:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

The value of e is approximately equal to 2.71828. We can get an approximate value of ***e***, by calculating only a partial sum of the infinite sum above (the more addends we add, the better approximation we get).

Implement the function:

`double eApprox(int n)`

This function is given a positive integer n, and returns an approximation of ***e***, calculated by the sum of the first (n+1) addends of the infinite sum above.

To test your function use the following main:

```
int main() {
    cout.precision(30);

    for (int n = 1; n <= 15; n++) {
        cout<<"n = "<<n<<'\t'<<eApprox(n)<<endl;
    }

    return 0;
}
```

Notes:

1. Pay attention to the running time of `eApprox`. An efficient implementation would run in $\Theta(n)$.
2. Since the values of the factorials will grow to be very large, use a variable of type `double` to store them.

Question 4:

a. Implement a function:
`void printDivisors(int num)`
This function is given a positive integer `num`, and prints all of `num`'s divisors in an **ascending order**, separated by a space.
For Example, if we call `printDivisors(100)`, the expected output is:
1 2 4 5 10 20 25 50 100

Implementation requirement: Pay attention to the running time of your function. An efficient implementation would run in $\Theta(\sqrt{num})$.

b. Use the function above when implementing a program that reads from the user a positive integer (≥2), and prints all it's divisors.
Your program should interact with the user **exactly** as it shows in the following example:
Please enter a positive integer >= 2: 100
1 2 4 5 10 20 25 50 100

Question 5:

Use the definition of Θ in order to show the following:

a. $5n^3 + 2n^2 + 3n = \Theta(n^3)$
b. $\sqrt{7n^2 + 2n - 8} = \Theta(n)$