Unit 9

Practice with Loops: Series/Summations

Series Approximations

- Many interesting real-valued functions or constants may be approximated as a rational number using a **series summation or product**
  - $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$
- Series are best generated using loops where each iteration generates one term and combines it with the previous terms (by adding or multiplying as necessary)

Simple Series

- Write a loop to generate the first *n* positive, odd numbers
  - Odd numbers: 1,3,5,7,9
- We could use two separate variables
  - An inductive/control variable to count to n and control how many repetitions
  - Another to produce the odd values
- It is more common to put the desired value in terms of the inductive/control variable
- If i ranges from 0 to n-1, then the first n odd numbers are generated by:
  - *2*i + 1*
- *Tip: Write a table of i and the desired value and try to see if a simply line (y = mx+b) can fit the data*

```
int n;
cin >> n;
int odd = 1;
for( int i=0; i < n; i++)
{
  cout << odd << endl;
  odd += 2;
}
```

Method 1: Generate the first n positive, odd numbers

```
int n;
cin >> n;
for( int i=0; i < n; i++)
{
  cout << _________ << endl;
}
```

Method 2: Generate the first n positive, odd numbers

Practice

- Write a loop to generate and output this sequence:
  - 3, 7, 11, 15, 19, 23, 27, 31, 35, 39
  - Trying doing so using only the inductive variable

```
for( int i=___; _________; _____)
{
  cout << _________ << endl;

}
```

- Write a loop to generate and output this sequence:
  - 0,0,1,1,2,2,3,3,4,4
  - Trying doing so using only the inductive variable

```
for( int i=___; _________; _____)
{
  cout << _________ << endl;

}
```

Another Example: Factorials

- Write a loop to compute n! (factorial)
  - $n! = 1 * 2 * \cdots * (n-1) * n = \prod_{i=1}^{n} i$
  - 0! is defined to just be 1
    - We would not want to multiply by 0 since any further multiplication would result in 0 as well

```
int n;
cin >> n;
int fact = ____;
for( int i=1; i <= n; i++)
{
   ____________________;
}
```

Calculating e^x

- Write a loop to generate the first n terms of the approximation of e^x
  - $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$
- Tips:
  - Generalize: Look at the pattern and write out the expression for the i-th term
  - Since 0! is a bit strange and just defined to be 1, pull out the first term and let the loop calculate the remaining terms
  - The first time around you can use the `pow(base, exp)` function; then try to see how you'd do it without using pow()
  - Keep a variable for i! updating it each iteration to be ready for the next

```
double x, e_x = ____;
int n, fact = 1;

cin >> x >> n;
for( int i=___; _________; _____)
{
   fact _______________;
   e_x _______________________;
}
```

Attempt 1

```
double x, e_x = ____, x_i = ____;
int n, fact = 1;

cin >> x >> n;
for( int i=___; _________; _____)
{
   x_i ______________________
   fact _______________;
   e_x _______________________;
}
```

Attempt 2