# CS2311 2020/2021 Mid-Term Quiz

1. Total time allowed: 90 minutes

2. Four questions and do not include any additional libraries

3. Download “Mid-Term.cpp” from Canvas, and re-name it using your student ID as “YourStudentID.cpp”, e.g., if your student ID is 1234567, name it as “12345678.cpp”

4. Write your student ID in “line 12” of “YourStudentID.cpp” as well

5. Write all your solutions in “YourStudentID.cpp”

**Q1 [Polynomial] (25%)**
Consider the following polynomial of degree **n** in **x**.

$$1 + x + x^2 + x^3 + \dots + x^n$$ with **n** (*data type:* ***int***) > 0 and **x** (*data type:* ***double***)

Write a C++ program according to the following requirements.

1. Enter **n.** *Assume that* **n** *is* already *an integer* number, but please check if **n** is **positive** and is not greater than **125**. If not, prompt user to enter again.
2. Output the polynomial. The character ^ can be displayed by "^" in cout.
3. Next, enter **x**. Check if **x** is within the range **[-250.0,125.0]**. If not, prompt user to enter again.
4. Evaluate the polynomial with the given **n** and **x**.

NOTE: Your code should **NOT** use the function **pow(x,n).**

**Expected Outputs (Inputs are underlined):**

**Example 1**

```
Enter n: 10
1+x^1+x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^10
Enter x: 25.5
Value = 1.20997e+14
```

**Example 2**

```
Enter n: 125
1+x^1+x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^10+x^11+x^12+x^13+x^14+x^15+x^16+x^17+x^1
8+x^19+x^20+x^21+x^22+x^23+x^24+x^25+x^26+x^27+x^28+x^29+x^30+x^31+x^32+x^33+x^34
+x^35+x^36+x^37+x^38+x^39+x^40+x^41+x^42+x^43+x^44+x^45+x^46+x^47+x^48+x^49+x^50+
x^51+x^52+x^53+x^54+x^55+x^56+x^57+x^58+x^59+x^60+x^61+x^62+x^63+x^64+x^65+x^66+x
^67+x^68+x^69+x^70+x^71+x^72+x^73+x^74+x^75+x^76+x^77+x^78+x^79+x^80+x^81+x^82+x^
83+x^84+x^85+x^86+x^87+x^88+x^89+x^90+x^91+x^92+x^93+x^94+x^95+x^96+x^97+x^98+x^9
9+x^100+x^101+x^102+x^103+x^104+x^105+x^106+x^107+x^108+x^109+x^110+x^111+x^112+x
^113+x^114+x^115+x^116+x^117+x^118+x^119+x^120+x^121+x^122+x^123+x^124+x^125
Enter x: 125.0
Value = 1.30991e+262
```

**Example 3**

```
Enter n: -5
Exponent out of range!
Enter n: 0
Exponent out of range!
Enter n: 8
1+x^1+x^2+x^3+x^4+x^5+x^6+x^7+x^8
Enter x: -250.001
Variable out of range!
Enter x: 125.00001
Variable out of range!
Enter x: 124.99
Value = 6.00469e+16
```

**Q2 [Longest Ascending Sequence in An Array] (25%)**

Please write a program to display the **longest continuous ascending** sequence out of **N** given numbers.

1. The program first reads **N** (type: **int**), which is the total count of input numbers (**N**<15).
2. The program then reads **N** numbers (type: **double**).
3. The program next prints the length of the longest continuous ascending sequence from the **N** input numbers. ("Ascending" means the next number should be straightly greater than the current one).
4. Finally print all the numbers in this sequence in 2 decimal places. Print a space after each number.
5. If there exist multiple sequences with the same length, print the first sequence.

Suppose there are five input numbers: 1.1, 2.2, -3.3, 4.4 and 5.5:

- Because 1.1<2.2, so the length of the sequence {1.1, 2.2} is two.
- Likewise, -3.3<4.4<5.5, so the length of the sequence {-3.3, 4.4, 5.5} is three.
- Three is greater than two, therefore the answer is {-3.3, 4.4, 5.5}.
- Note: sequence {2.2, -3.3} is not considered as it is not ascending.
- Note: sequence {-3.3, 4.4} is not considered as it could be longer by including 5.5.
- Note: sequence {1.1, 2.2, 4.4, 5.5} is not considered as it's not continuous due to -3.3.

Hint: because the result is a continuous sequence, you can declare *vIdx* and *vLen* to store the first element's index and the length of the longest continuous ascending sequence obtained so far in the loop, respectively.

**Expected Output (Inputs are underlined):**

**Example 1**

```
Input the count of number(s) N: 5
Input 5 number(s): 1.1 2.2 -3.3 4.4 5.5
Length of the longest ascending sequence is: 3
-3.30 4.40 5.50
```

**Example 2**

```
Input the count of number(s) N: 4
Input 4 number(s): 1.1 2.2 -3.3 4.4
Length of the longest ascending sequence is: 2
1.10 2.20
```

**Example 3**

```
Input the count of number(s) N: 3
Input 3 number(s): 777 777 777
Length of the longest ascending sequence is: 1
777.00
```

**Q3 [Number Searching] (25%)**

Given two **non-negative** integers **A** (*data type: **int***) and **B** (*data type: **int***), write a C++ program to determine whether **A** *contains* **B**, wherein "contains" means an exact match. For example,

- If **A** is 12345 and **B** is 234, **A** contains **B**, where the matching part is underlined in **A**.
- If **A** is 12345 and **B** is 235, **A** does **not** contain **B**, even **A** has the digits 2, 3 and 5. It is not an exact match.

Notes:

1. Assume both inputs are **correct**. **No need** to check the correctness of input.
2. **A** and **B** are both in the simplest form (e.g., no **0** prefix such as **001**).
3. Students are **not** supposed to use *char*, *string* or *libraries* other than **cout**.

Hint: Suppose **A** is 12345 and **B** is 234. Because B has **n** (e.g., 3) digits, you can check the last **n** (e.g., 3) digits of **A**. If these **n** (e.g., 3) digits do not equal to B, you can update **A** by discarding its last digit, e.g., **A** is updated to 1234, and continue to check the last **n** digits of this new **A**.

In addition, when A equals to B, this can be considered as a special case to handle.

**Expected Outputs (Inputs are underlined):**

| |
|---|
| **Example 1** |
| Input A and B separated by space: **12345** **234**<br>12345 contains 234 |
| **Example 2** |
| Input A and B separated by space: **12345** **235**<br>12345 does not contain 235 |
| **Example 3** |
| Input A and B separated by space: **54321** **54321**<br>54321 contains 54321 |

**Q4 [Shape Printing] (25%)**
Please write a program that read a positive integer number (data type: **int**) as input, and first print the sum **N** of the **largest even** digit and the **smallest odd** digit. For example,

- If the input is 21342, the output **N** is 5, because 4 + 1 = 5.
- If the input is 26842, the output **N** is 8, because 8 + 0 = 8 (no odd digit in the input)
- If the input is 23366, the output **N** is 9, because 6 + 3 = 9 (count each digit only once)

Notes:

- You may not know the number of digits from the input in advance.
- We assume the input is valid, i.e., no need to check the correctness of input.
- We assume the input is in the range that an int-type variable can represent.

Next, print a reversed tree controlled by the sum **N** obtained above:

- The number of triangles in the tree is **N**, e.g., **N** is 2 in the table below.

- In the $i$-th triangle (counted from bottom to top), the number of rows is $i$ + 1, e.g., the bottom ($i$ = 1) triangle in the table below has 2 (= 1 + 1) rows.
- In the $i$-th triangle (counted from bottom to top), its top row contains $2 \times i + 1$ asterisks (" $*$ "), and the bottom row contains one asterisk.

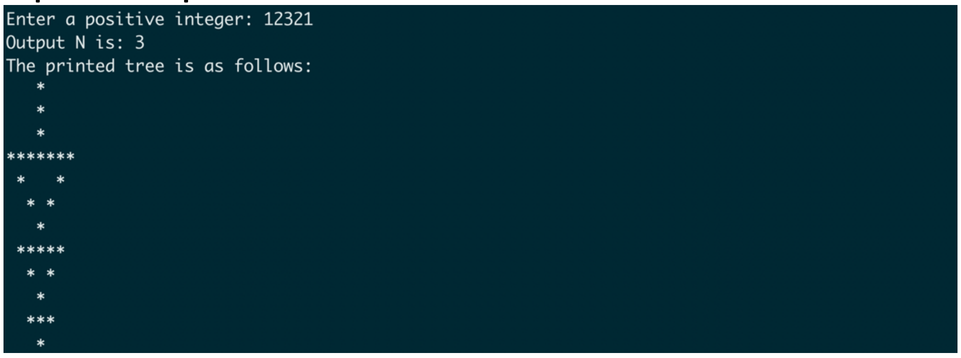
- Except the top row and bottom row of each triangle, spaces are filled in between two asterisks (" $*$ ") at the beginning and the end in each of rest rows (if any).
- There is no space before the first asterisk on the top row of the top triangle.

- Above the top triangle, there is a trunk alone the center line of the tree. The height (number of rows) of the trunk is also **N**, e.g., this height is 2 in the table below.

**Table:** illustrating the tree when **N** is 2

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | * | | | | | | | | | | | | |
| | | * | | | | | | | | | | | | |
| * | * | * | * | * | | | | | | | | | | |
| | * | | * | | | | | | | | | | | |
| | | * | | | | | | | | | | | | |
| | * | * | * | | | | | | | | | | | |
| | | * | | | | | | | | | | | | |

**Expected Output:**



```
Enter a positive integer: 12321
Output N is: 3
The printed tree is as follows:
   *
   *
   *
*******
 *   *
  * *
   *
 *****
  * *
   *
  ***
   *
```