CS317 Homework 1

Due Date: Check Canvas for Due Date

Total Points: 120

Submission Instructions: You have two options. Either you can submit the paper copy in class on the due date or you can upload it on the Canvas. If you chose to upload it on the Canvas, it must be a single PDF file.

1) Draw a flowchart of Algorithm Design and Analysis Process. (5 points)

2) What is a Basic Operation? Why do we need to count basic operations? Give few examples of basic and non-basic operations. (5 points)

3) Arrange the running time of functions mentioned below by the order (increasing) of their growth. (5 points)

n^3, n, nlog(n), log(n), 2^n, n^2, n!

4) Solve the following problems. (10 points)

#1) Use the most appropriate notation among O, Θ, and Ω to indicate the time efficiency class of sequential search

a. in the worst case.

b. in the best case.

c. in the average case.

#2) Use the informal definitions of O, Θ, and Ω to determine whether the following assertions are true or false.

a. $n(n+1)/2 \in O(n^3)$

b. $n(n+1)/2 \in O(n^2)$

c. $n(n+1)/2 \in \Theta(n^3)$

d. $n(n+1)/2 \in \Omega(n)$

5) What the three asymptotic notations that are used to compare and rank the order of growth of algorithms? Explain them **in plain English** with one example each. (5 points)

6) Give the list of basic asymptotic efficiency classes. (5 points)

7) Give the general plan for analyzing the non-recursive algorithms. (5 points)

8) Give the general plan for analyzing the recursive algorithms. (5 points)

9) Consider the following 3 algorithms and answer the (a) to (d) questions for each algorithm. (15 points)

ALGORITHM *Mystery*(n)

```
//Input: A nonnegative integer n
S ← 0
for i ← 1 to n do
    S ← S + i * i
return S
```

ALGORITHM *Secret*$(A[0..n-1])$

```
//Input: An array A[0..n − 1] of n real numbers
minval ← A[0]; maxval ← A[0]
for i ← 1 to n − 1 do
    if A[i] < minval
        minval ← A[i]
    if A[i] > maxval
        maxval ← A[i]
return maxval − minval
```

ALGORITHM *Enigma*$(A[0..n-1, 0..n-1])$

```
//Input: A matrix A[0..n − 1, 0..n − 1] of real numbers
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if A[i, j] ≠ A[j, i]
            return false
return true
```

a. What does this algorithm compute?

b. What is its basic operation?

c. How many times is the basic operation executed?

d. What is the efficiency class of this algorithm?

10) Give the pseudo code of the factorial function F (n) = n! Analyze this algorithm, i.e. setup a recurrence relation and solve it to find the running time in Θ notation. (10 points)

11) Design an algorithm (write in pseudo code) which takes less than $\Theta(n)$ to search an entry (name of a person) in a telephone directory. Analyze this algorithm for its worst-case input situation. Make necessary assumptions to simplify your comparisons. If you don't know what is a telephone directory, check out this link https://en.wikipedia.org/wiki/Telephone_directory (10 points)

12) Design an $\Theta(n)$ algorithm to determine if the given array is already sorted or not. Analyze this algorithm for its worst-case input. Setup the summation and solve it. (10 points)

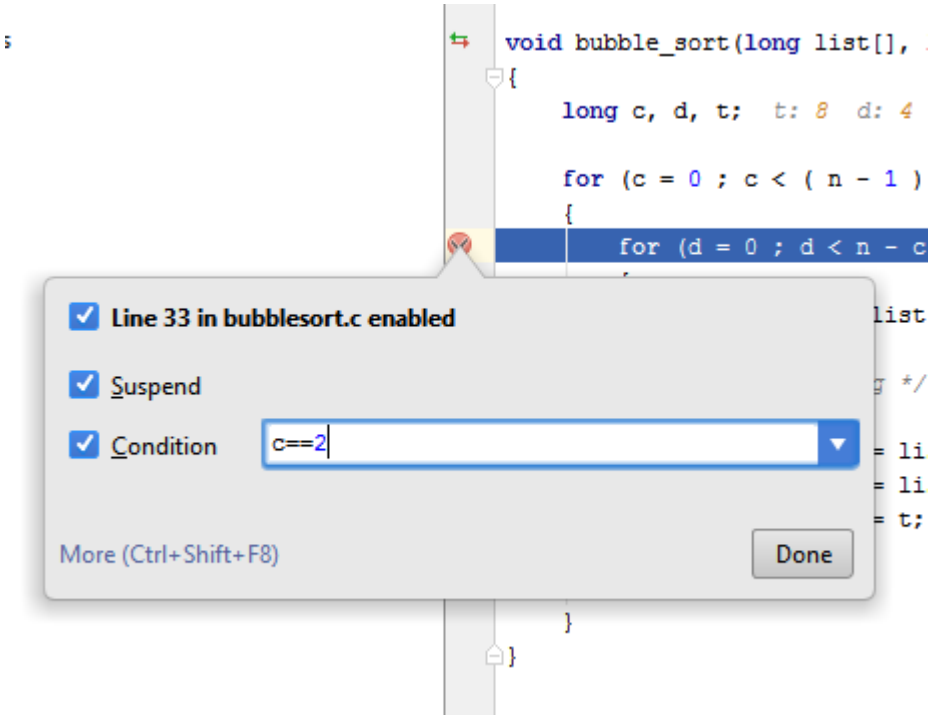
13) Write down the pseudo code of any decrease-and-conquer type sorting algorithm and analyze the running time of this algorithm for its worst-case input scenario. Also mention the running time of this algorithm for its best and average case input scenarios. (10 points)

14) Write a BubbleSort program using the language of your choice to sort the given list and demonstrate the understanding of debugging techniques using any modern IDE such as JetBrains or Microsoft Visual Studio. (10 points)

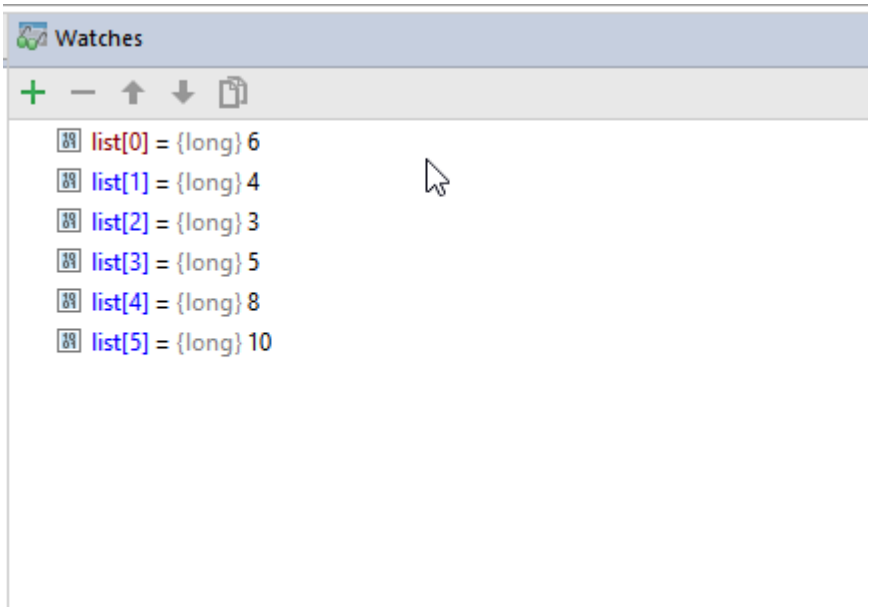
Instructions: Use the following numbers as input to your BubbleSort program.

6, 8, 10, 4, 3, 5

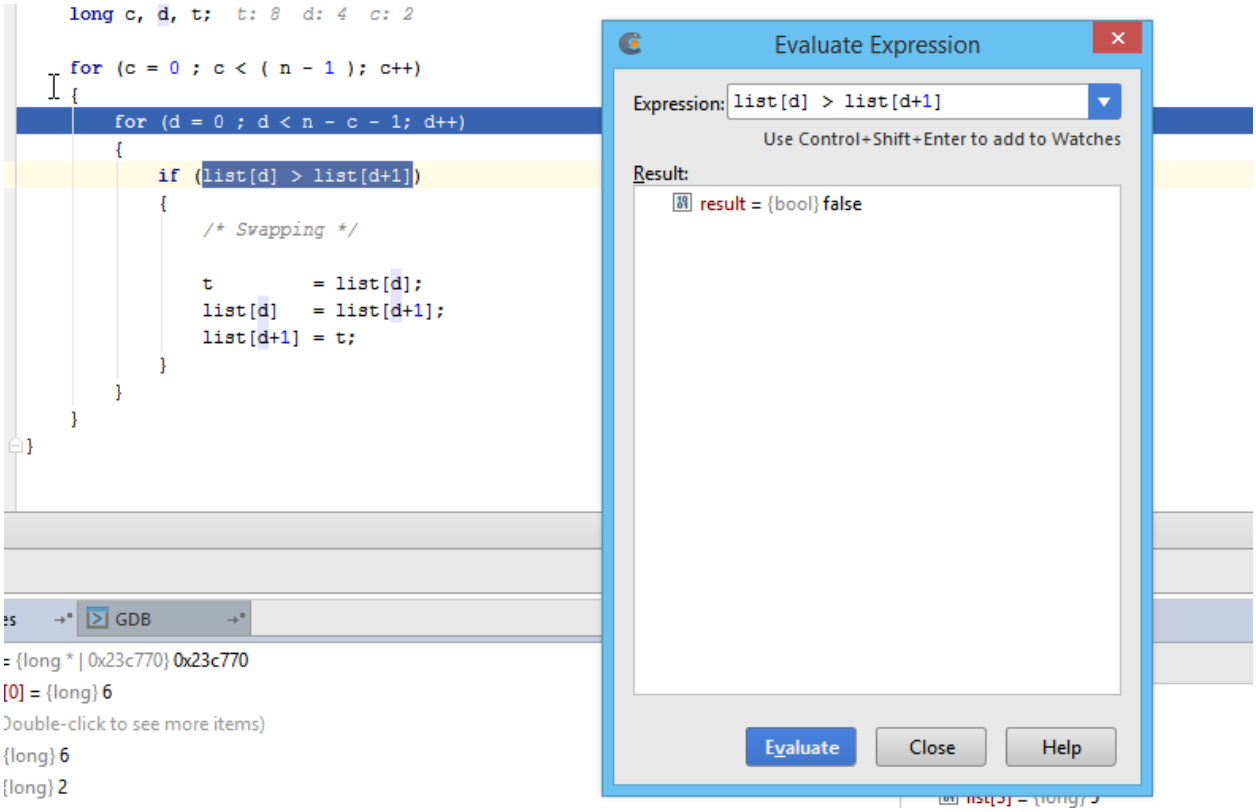
Bubble Sort algorithm uses two loops. Set a conditional break point as shown below at the inner most loop such that it will stop after 3 passes of the outer loop are completed.



Once your program stops after completion of 3 passes, show the current state of your input array using a watch window. That is showing a partially sorted array as below:



Now, use an “Evaluate Expression” or “Quick Watch” feature of the IDE you are using to evaluate the basic operation of the Bubble sort algorithm.



Deliverables:

- Your program's source code and output (2 points)
- Fours Screenshots (8 points):
  - 3 screenshots as shown above
  - one screenshot of the complete IDE window.

15) Convert the "Brute Force Password Generation" program covered in the class from C# to your favorite language. For this particular program, you can use any programming language such as Ruby, Java, Python, C++ etc., but you cannot use C# as the program is already given in C#. Run this program for password of lengths 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30 and 35 etc. Record the execution time of this program in milliseconds for each of this password length using **Empirical or Experimental Analysis**

technique shown in the class. Record the execution time in the table below. (10 points)

Notes:

- You do not need to convert the program line by line. You are allowed some flexibility/creativity on it.
- If you are using C++, the BigInteger data type is not available by default. Please see the note on the next page about using cpp_int type from Boost library.

Password Length	Execution time in milliseconds to generate all the combinations
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
15	
20	
25	
30	
35	

Deliverables:

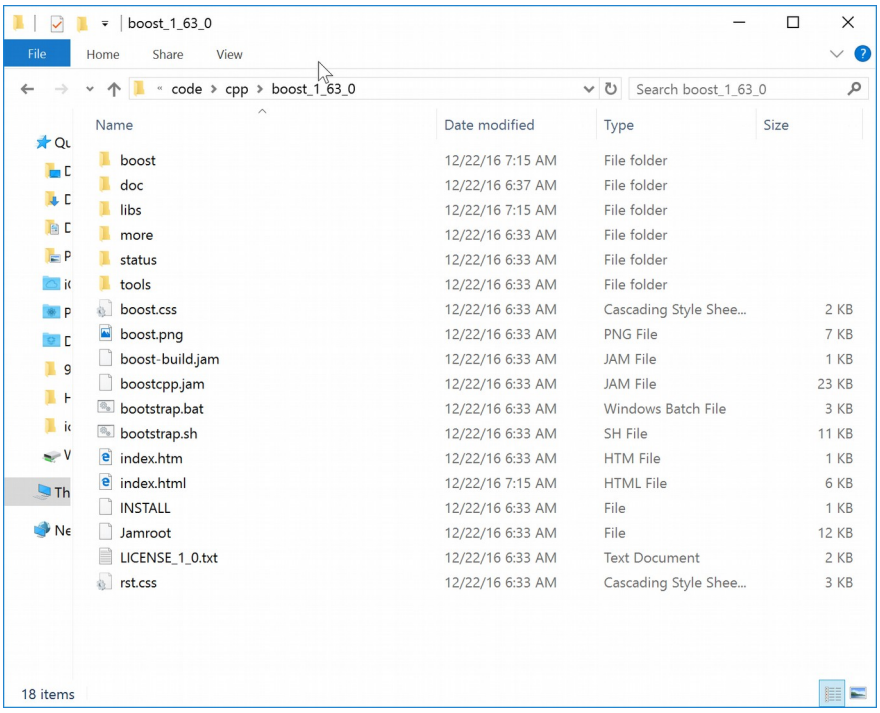
- Your program's source code
- Above table

Note on how to use BigInteger data type in C++

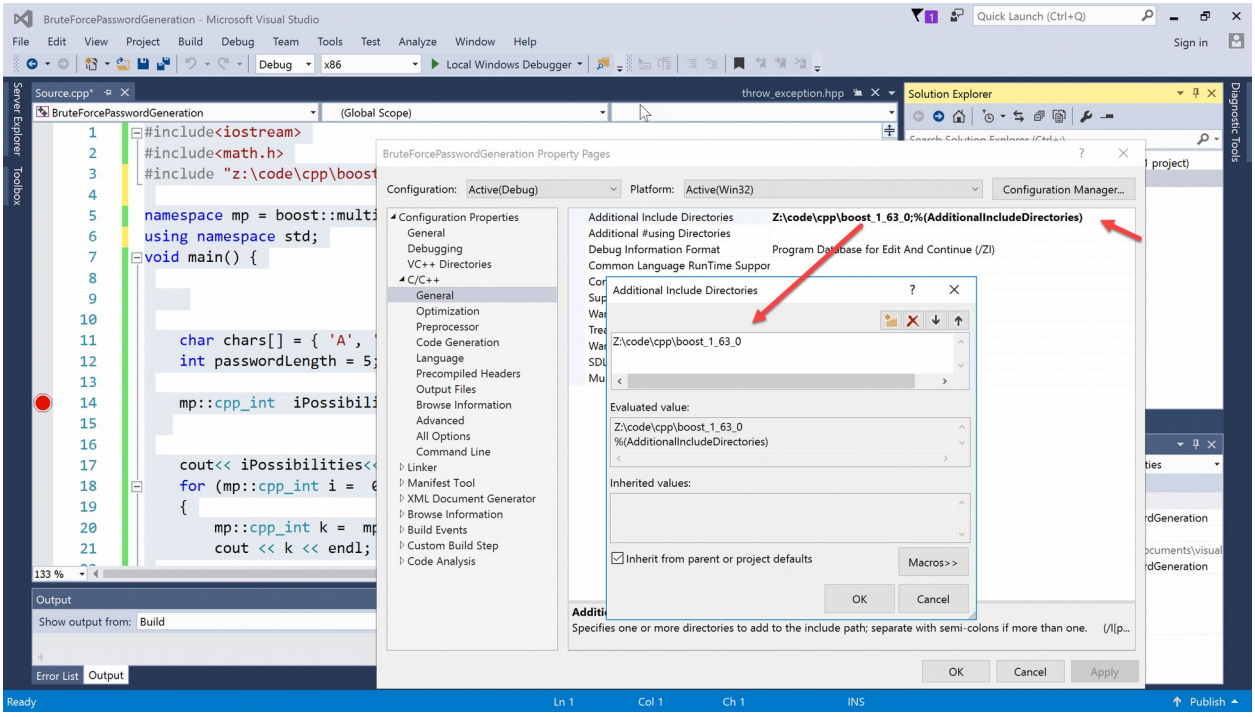
To store big integers in C++, you can use the Boost library (http://www.boost.org/). Boost provides **cpp_int** data type that can be used to store big integers.

You can download the Boost library from https://sourceforge.net/projects/boost/files/boost/1.63.0/

Once you have downloaded the Boost library, extract it in a folder as shown below.



Then add this folder under “Additional Include Libraries” in your Visual Studio project settings as shown below. You can find more instructions here http://www.boost.org/doc/libs/1_63_0/more/getting_started/windows.html#link-from-within-the-visual-studio-ide



Here is the example program using cpp_int data type from the Boost library -

```
#include<iostream>
#include<math.h>
#include "z:\code\cpp\boost_1_63_0\boost\multiprecision\cpp_int.hpp"

namespace mp = boost::multiprecision;
using namespace std;
void main() {



	char chars[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };
	int passwordLength = 5;

	mp::cpp_int  iPossibilities = (mp::cpp_int)pow(26,  passwordLength);


	cout<< iPossibilities<< " words total. ";
	for (mp::cpp_int i =  0; i < iPossibilities; i=i+1)
	{
		mp::cpp_int k =  mp::cpp_int(i % 26);
		cout << k << endl;

	}
}
```