CSDS 233 Assignment 1 Due: September 25, 2023, 11:59PM

General instruction: This assignment includes two parts, written and programming. Please write/type your answers neatly so they can be readable. Please submit a single **PDF** file for the written part and a **zip** file for the programming part **before 11:59 P.M. on September 25, 2023**. File name format: P1_YourCaseID_YourLastName.zip (or pdf).

The special office hour for this assignment will be from **6:00 to 7:00 P.M. on September 20** in Zoom. You can also send an email to wxy215@case.edu for written problems and ajn98@case.edu for coding problems.

Written Problems (50 pts)

P1. Simplify (as much as possible) the following Big-O notation and explain. (15 pts)

1. $O(log_2 n^2 + (log_2 n)^2 + log_2 n)$
2. $O(n^2 + (n + 1)^2 + (n/2)^2)$
3. $O(\sqrt[3]{n} + log_2 n)$
4. $O(1 + 2 + 3 +... + 1000)$
5. $O(1 + 3 + 5 +... + (2n + 1))$

P2. Provide a tight Big-O notation and explain for each part of the pseudocode. (23 pts)

1. (2 pts)
```
int y = 0;
for (int i = 1; i < n; i *= 2) {
    y++;
  }
```

2. (3 pts)
```
int y = 0;
for (int i = 0; i < n; i++) {
  for (int j = n; j > i; j--) {
    y++;
  }
}
```

3. (5 pts)
```
int y = 0;
    for (int i = 1; i < n; i ++) {
      for (int j = 1; j<i*i; j++){
        y++;
      }
    }
```

4. (5 pts)
```
 int y = 0;
    for (int i = 1; i < n; i ++) {
      for (int j = 1; j<sqrt(i); j++){
        y++;
      }
    }
```

5. (8 pts)
```
int y = 0;
if (x > 0) { // x is a random number
    for (int i = n; i > 1; i--) {
        for (int j = i; j > 1; j = j / 3) {
            y++;
        }
    }
} else {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n / (i + 1); j++) {
            y++;
        }
    }
}
```

P3. In the context of array lists, linked lists, and doubly linked lists, explain their worst-case big O time complexities for the following operations. You need to provide answers for all 4 operations across all 3 types of lists, totaling 12 in all. (12 pts)

1. Finding an element in the list based on its value
2. Inserting an element at the beginning of the list
3. Removing an element from the end of the list
4. Inserting an element at the middle of the list

Programming Exercise (50 pts)

1. Write a program in iterative and recursive approaches that returns a boolean whether a user input string is a palindrome (https://en.wikipedia.org/wiki/Palindrome). Method names should be palindromeIterative(String input) and palindromeRecursive(String input) respectively. Call the function that is more efficient in terms of Time Complexity and write its Big-O notation in comments above each method.

2. Building off the previous program, implement the below additional functions (use the specified function names below, but input variable names are at your discretion as long as they are meaningful:
   - **boolean anagramChecker(String x, String y)**: Returns true if one string is the anagram of another string ("silent" is an anagram of "listen" and vice versa)
   - **String addSubstring(String input, String substring, int index)**: Returns the substring added to the input string given a specified index
   - **int getLength(String input)**: Returns the length of a given input string
   - **int occuranceCounter(String input, String substring)**: Returns the number of occurrences a specified substring is in the input string
   - **String senetenceReversal(String input)**: Returns only the sentence string in reverse order of words (input string is "This is a test." which gives the return string, "test a is This.")

Additional requirements:

The program should continuously run with the user presented with an option menu upon start up. The option menu should print the options of the functions above, the palindrome check, and a quit option which stops the program. An acceptable output upon start up, calling the **addSubstring()** function, and then quitting would be:

```
Welcome to the App
  1. Palindrome Check
  2. Anagram Check
  3. Add Substring
  4. Get Length
  5. Count Occurances
  6. Reverse Sentence
  7. Quit
Choose an option: 3

Enter string: Hello
Substring to be inserted: World!
Index placement: 5

New string: HelloWorld!

Choose an option: 7

Have a nice day:)
```

Grading:

- Part 1: 15 pts
  - Iterative palindrome implementation: 5 pts
  - Recursive palindrome implementation: 5 pts
  - Big-O/function: 2 pts
  - Calling most efficient method: 1 pts
- Part 2: 25 pts
  - Each function implemented correctly: 4 pts
  - Continuous user input: 5 pts
- Design & Style (consistent function/variable names & clean comments): 10 pts
  - Style: 2 pts
  - Design: 2 pts
  - Comments: 6 pts
- Please comment any additional resources used and/or contributors used