CS472 Assignment #2

This assignment has 6 problems, for a total of 100 points.

Submission Instructions
Please attach a PDF document with your answers to these questions as an attachment to your submission in Blackboard. Handwritten submissions are acceptable if scanned in a readable fashion into a PDF document. Points will be deducted if you do not follow these instructions.

1. (10 points) You are given the following algorithm:

```
Algorithm 1: FunkyIndexes
  Input: Some value $n > 0$
  Output: A summation based on the value of $n$
  r = 0;
  for $i$ = 1 to $n$ do
      for $j$ = 1 to $i$ do
          for $k$ = $j$ to $i+j$ do
              r = r + 1;
  return r
```

   What value is returned by this algorithm, expressed as function of n? What is the $O(n)$ of this algorithm? Justify your answer.

2. (10 points) Show directly that $f(n) = n^2 + 3n^3 \in \Theta(\mathrm{n}^3)$. That is, use the definitions of O and Ω to show that $f(n)$ is in both $O(n^3)$ and $\Omega(n^3)$.

3. (10 points) The lectures contain a proof that $2^{n+1} \rightarrow \Theta(2^n)$. Can you generalize this thought? Is $a^{n+1} \rightarrow \Theta(a^n)$ if a is constant? Justify your answer.

4. (10 points) Consider the following algorithm that checks whether a graph represented by its adjacency matrix is complete:

```
Algorithm 2: GraphComplete
  Input: An zero-indexed adjacency matrix A of a undirected graph G
  Output: A Boolean value of true if graph is complete, false otherwise
  if $n$ = 1 then
      return true;
  else
      if NOT (GraphComplete (A[n-2],A[n-2])) then
          return false;
      else
          for $j$ in [0..n-2] do
              if A[n-1,j] = 0 then
                  return false;
```

   What is this order of this algorithm in the worst case? Justify your answer.

5. (30 points) The Tower of Hanoi problem is often used to illustrate that there is more than one way to skin a cat; in other words, there are many ways of solving a problem.

   Remember Gray codes from your Digital Logic class? Mathematically, an n-bit Gray code is a 1-1 mapping from the range $[0 \ldots (2^{n-1}) - 1]$ s.t. the binary representation of consecutive numbers vary by exactly one bit position. This codes from mechanical engineering where a physical device called a "shaft encoder" encodes 2^n different angles on a disk that a photo-receptor can read to determine a value. Because of the hardware design, you have to minimize the change in number of bits to 1 in the encoding.

   You can generate Gray codes using a recurrence relation:

$$\begin{aligned}
G_1 &= [0, 1] \\
G_2 &= [0G_1, 1\overline{G_1}] = [00, 01, 11, 10] \\
G_3 &= [0G_2, 1\overline{G_2}] \\
&= [000, 001, 011, 010, 110, 111, 101, 100] \\
G_n &= [0G_{n-1}, 1\overline{G_{n-1}}]
\end{aligned}$$

   where $\overline{G_n}$ is G_n is reverse order.

   So...here's a C++ function that generates a vector of strings containing all n bit Gray codes:

```
vector<string> generateGray(int n) {
     // Base case
     if (n <= 0)
         return {"0"};
     if (n == 1)
     {
       return {"0","1"};
     }
     //Recursive case
     vector<string> recAns= generateGray(n-1);
     vector<string> mainAns;
     // Append 0 to the first half
     for(int i=0;i<recAns.size();i++) {
       string s=recAns[i];
       mainAns.push_back("0"+s);
     }
      // Append 1 to the second half
     for(int i=recAns.size()-1;i>=0;i--) {
        string s=recAns[i];
        mainAns.push_back("1"+s);
     }
     return mainAns;
}
```

   For any number of n disks stuck on the Tower of Hanoi, one can determine what disk to move by counting from 0 to $n - 1$ in Grey Code. Remembering that Gray Codes are organized so that only one bit position changes as numbers increase, one can determine which disk to move by looking at the bit position that changes as the count increases. How do you know where to move the disk? For the smallest disks, there are two possibilities but always only one for any other disk. For the smallest disk. let f by the starting peg, t be final peg, and r the remaining peg. If the number of disks is odd, the smallest disks cycles along the pegs $f \to t \to r$ or, if even, $f \to r \to t$.

   Write an algorithm that uses Gray Codes to solves the Tower of Hanoi problem for any number of disks n. **NOTE**: I am asking for you to write an algorithm, not go and break out the C++ compiler or the Python interpreter (or, heaven forbid, the COBOL compiler).

6. (30 points) The authors of modern software libraries are nice enough to include a `sort()` function for use in applications. These libraries claim that the sort functions use an algorithm that is at least $O(n * lg(n))$ in nature. Let's test this claim by writing a small program.

   In your language of choice, use the sort routine provided by the language to sort randomly generated data sets of 5, 10, 50, 100, 500, 1000, 5000, and 10,000 integers. Since we are looking at average data set sizes, you will need to sort at least 10 or more random data sets for each data set size.

   You should implement an algorithm that looks like this:

   Algorithm 3: Test the order of the library sort

```
Output: A data set of run times for each data set size
for size ∈ [5, 10, 50, 100, 500, 1000, 5000, 100000] do
    Set totalRunTime ← 0;
    for runNumber ∈ [0 .. NumberOfRuns) do
        GenerateRandomDataSet (dataSet,size);
        Note start time;
        Sort (dataSet);
        Note end time;
        Set elapsed time to difference between end and start times;
        Add elapsed time to totalRunTime;
    Set averageRunTime = totalRunTime / NumberOfRuns;
```

   The function `GenerateRandomDataSet(dataSet, size)` needs to generate a random set of integers of whatever size is passed into the function as an input.

   Create a graph (using Excel or your favorite graphing tool) that graphs data set size versus the average run time for each size. Include a plot of data set size versus $f(n) = n * lg(n)$ as well. Submit your source code, graph, and a short (1-3 paragraph) analysis of the results (each of which contributes 10 points to the grade for this problem).

   Hint: you will need to research two items for this question: (1) the functions in your language of choice to deal with function timing, and (2) how to call the `sort()` function in your language of choice. If you are programming in C++, review the information found at `https://en.cppreference.com/w/cpp/numeric/random` for more information on random number generation and `https://en.cppreference.com/w/cpp/algorithm/sort` for sorting in C++.