CptS 223 – Advanced Data Structures in C++

Written Homework Assignment 2: Big-O and Algorithms

I. Problem Set:

1. **(20 pts)** Given the following two functions which perform the same task:

int g (int n)
{
  if(n <= 0)
  {
  return 0;
  }
  return 1 + g(n - 1);
}
	int f (int n)
	{
	  int sum = 0;
	  for(int i = 0; i < n; i++)
	  {
	  sum += 1;
	  }
	  return sum;
	}

a. (10 pts) State the runtime complexity of both f() and g().

Runtime of f() would be O(n) because it would end after being called n times. This will run from i=0 to i=n-1
Runtime of g() would be the same if not similar, as it would also complete after it runs n times through, thus it would be O(n). This will run from n=0 to n=n, making f() just a bit faster.

b. (10 pts) Write another function called "int h(int n)" that does the same thing but is significantly faster.

```
Int h(int n)
{
        return n*(n-1)/2;       ← this function takes O(1) time, much faster.
}
```

2. **(15 pts)** State g(n)'s runtime complexity:

```
int f (int n){
if(n <= 1){
      return 1;
   }
   return 1 + f(n/2);
}

int g(int n){
   for(int i = 1; i < n; i *= 2){
f(n);
   }
}
```

g()s runtime complexity is O(log n) because it forms a series by iteration. And we end up with 1+(log(n)) but the 1 is meaningless as we only care about the log(n).

3. **(20 pts)** Write an algorithm to solve the following problem (10 pts)

Given a nonnegative integer n, what is the smallest value, k, such that

$$1n,\ 2n,\ 3n,\ \ldots,\ kn$$

contains all 10 decimal numbers (0 through 9) at least once? For example, given an input of "1", our sequence would be:

$$1 * 1, 2 * 1, 3 * 1, 4 * 1, 5 * 1, 6 * 1, 7 * 1, 8 * 1, 9 * 1, 10 * 1$$

and thus k would be 10. Other examples:

Integer Value	K value
10	9
123456789	3
3141592	5

Make an array j with 0-9
Cin >> number n

Read next digit in number n
Check digit == j[i]

(Iteration)
When digit == j[i]
Store in a checklist
Repeat the steps above until no more digits

If no more digits to read and list is not full
Multiply n by k (k = 1, k++)
Then Repeat steps for checking the numbers
Then repeat multiply until checkList is full

When checklist full
Output k

(10 pts). Can you directly formalize the worst case time complexity of this algorithm? If not, why?

```
O(k*number of times loops)
```

4. **(20 pts)** Provide the algorithmic efficiency for the following tasks. Justify your answer, often with a small piece of pseudocode to help with your analysis.

    a. (3 pts) Determining whether a provided number is odd or even.

```
O(1)

If num divisible by 2

Even

Else

odd
```

    b. (3 pts) Determining whether or not a number exists in a list.

```
O(n)

List = L
Number = num
I = integer

If I = num
In list

Else
Not in list
```

    c. (3 pts) Finding the smallest number in a list.

```
O(n)

Arr list (1-10)
Int I;
Cin >> i

If A[i] < min
I is min (smallest number)
```

d. (4 pts) Determining whether or not two **unsorted** lists of the same length contain all of the same values (assume no duplicate values).

```
O(n^2)

Arr j
Arr k

Set first value of j equal to all values in k
Iterate through until match found

If arr j size == arr k size
Same size
```

e. (4 pts) Determining whether or not two **sorted** lists contain all of the same values (assume no duplicate values).

```
O(n)

Arr j

Arr k

For i=1 i++ to n

If (j[i] == k[i])

      Keep checking

Else

Not same values
```

f. (3 pts) Determining whether a number is in a balanced BST.

```
O(log n)

Int i
Number = num

Check if num = I
If not go down the tree and check for left and right nodes
Check left,

if not equal
Check right
```

```
Once there are not any child nodes, number not found (if num was not found above)
```

5. **(25 pts)** Write a pseudocode or C++ algorithm to determine if a string `s1` is an *anagram* of another string `s2`. If possible, the time complexity of the algorithm should be in the worst case O(n). For example, 'abc' - 'cba', 'cat' - 'act'. s1 and s2 could be arbitrarily long. It only contains lowercase letters a-z. Hint: the use of histogram/*frequency* tables would be helpful!

```
Anagram check
 Iterator
Count = s1[i]++
Count = s2[i]--

If strings have different length
Output false

Iterator i = 0; i < Count
       If count[i] = false
Else true

If anagram check == true output s1 = s2

Else output s1 != s2

Complexity: O(n) by using one count array, we increment up and down and if count
values are 0 then s1 == s2
```

II. Submitting Written Homework Assignments:

1. On your local file system, create a new directory called HW2. Move your HW2.pdf file into the directory. In your local Git repo, create a new branch called HW2. Add your HW2 directory to the branch, commit, and push to your private GitHub repo created in PA1.
2. Do not push new commits to the branch after you submit your link to Canvas otherwise it might be considered as late submission.
3. Submission: You must submit a URL link of the branch of your private GitHub repository to Canvas.