CSCI-104 Written Homework #1

Problem 1: Runtime Analysis

In Big-Θ notation, analyze the running time of the following four pieces of code/pseudo-code. Describe it as a function of the input (here, n).

Part (a)

```
void f1(int n)
{
    int i=2;
    while(i < n){
        /* do something that takes O(1) time */
        i = i*i;
    }
}
```

- Since i multiply itself every iteration, it grows in the rate of 2^{2^k} after k^{th} iteration. Also, because $i \leq n$, by solving $2^{2^i} = n$, we can get $i \leq \log\log n$, so the answer is $\Theta(\log\log n)$.

Part (b)

```
void f2(int n)
{
    for(int i=1; i <= n; i++){
        if( (i % (int)sqrt(n)) == 0){
            for(int k=0; k < pow(i,3); k++) {
              /* do something that takes O(1) time */
            }
        }
    }
}
```

- The inner loop takes $\Theta(i^3)$ time, and the inner loop only executes when `i % (int)sqrt(n)) == 0` , which happens at most n times. Thus, overal it is

$$\Theta(n + ((\sqrt{n})^3 + (2\sqrt{n})^3 + \cdots + (\sqrt{n}\sqrt{n})^3))$$

$$= \Theta(n + ((\sqrt{n})^3 \cdot (1^3 + 2^3 + \cdots + (\sqrt{n})^3)))$$

$$= \Theta(n + ((\sqrt{n})^3 \cdot \frac{n^2 + n + 2n\sqrt{n}}{4}))$$

$$= \Theta((\sqrt{n})^3 \cdot n^2)$$

$$= \Theta(n^{\frac{7}{2}})$$

Part (c)

```
for(int i=1; i <= n; i++){
  for(int k=1; k <= n; k++){
    if( A[k] == i){
      for(int m=1; m <= n; m=m+m){
        // do something that takes O(1) time
        // Assume the contents of the A[] array are not changed
      }
    }
  }
}
```

- The outer loop has runtime of $\Theta(n)$, and the inner loop has runtime of $\Theta(\log n)$ since m doubles every iteration.
- For the middle loop and the `if` condition, through out the execution, at most n times the `if` statement is true, as it is finding how many numbers in `A[]` is between 1 and n.
- Thus, the total time complexity is the times that the `if` statement is true times the complexity of inner loop: $n \cdot \Theta(\log n) = \Theta(n \log n)$.

Part (d)

```
int f (int n)
{
  int *a = new int [10];
  int size = 10;
  for (int i = 0; i < n; i ++)
     {
        if (i == size)
          {
             int newsize = 3*size/2;
             int *b = new int [newsize];
             for (int j = 0; j < size; j ++) b[j] = a[j];
             delete [] a;
             a = b;
             size = newsize;
          }
        a[i] = i*i;
     }
}
```

- The outer loop has a complexity of $\Theta(n)$, and in the inner loop, it performs a conditional sizing that if the the index is out of range and resize to 1.5x the previous size.
- The number of times of resizing is proportional to $\log n$ and thus the total complexity of resizing will be $\sum_{i=0}^{\log n} 10 \cdot \left(\frac{3}{2}\right)^i = 10\Theta\left(\left(\frac{3}{2}\right)^{\log n}\right) = \Theta(n)$.
- Overall, the runtime combined with the outer and inner loop is $\Theta(n)$.

Problem 2: Linked List Recursion Tracing

Question a

What linked list is returned if llrec is called with the input linked lists `in1 = 1,2,3,4` and `in2 = 5,6` ?

Answer:

The Function Execution in the following order:

1. **First Call:** `llrec(1, 5)`
   - Set `1 -> next` to the result of `llrec(5, 2)` .
2. **Second Call:** `llrec(5, 2)`
   - Set `5 -> next` to the result of `llrec(2, 6)` .

3. **Third Call:** `llrec(2, 6)`
   - Set `2 -> next` to the result of `llrec(6, 3)`.
4. **Fourth Call:** `llrec(6, 3)`
   - Set `6 -> next` to the result of `llrec(3, nullptr)`.
5. **Fifth Call:** `llrec(3, nullptr)`
   - `in2` is `nullptr`, so return `in1 = 3`.

   After this call:
   - `6 -> next` points to `3`.
   - `2 -> next` points to `6`, which points to `3`.
   - `5 -> next` points to `2`, which points to `6 -> 3`.
   - `1 -> next` points to `5`, which points to `2 -> 6 -> 3`.
6. **Final Answer:** After the Fifth Call, the returned linked list `in1 = 1 -> 5 -> 2 -> 6 -> 3`.

Question b

What linked list is return if llrec is called with the input linked lists `in1 = nullptr` and `in2 = 2`?

Answer: the programm will go throgh the first `if` statement and return `in2` which is `2`.