CSCI 104 HW 1

Problem 1: Runtime Analysis

Part a)

```
int i = 2;
while(i < n);
    ...
    i = i * i
}
```

	i	i<n
After Iteration 0 ->	2	2<n
1 ->	4	4<n
2 ->	16	16<n
3 ->	256	256<n

in iteration 1: $2 \cdot 2 = 2^2 = 4 = 2^{2^1}$

in iteration 2: $4 \cdot 4 = 4^2 = 16 = 2^{2^2}$

in iteration 3: $16 \cdot 16 = 16^2 = 256 = 2^{2^3}$

$i = 2^{2^Y}$

$(2^3)^2$

- Loop ends when $i < n$, so to find the number of iterations, Y, substitute, $2^{2^Y} < n$, then $\log_2(2^{2^Y}) < \log_2(n)$

$= 2^Y < \log_2(n)$

$= \log_2(2^Y) < \log_2(\log_2(n))$

$= Y < \log_2(\log_2(n))$

- Since inside the loop other things do O(1) time, then runtime of this code is $\boxed{\Theta(\log_2(\log_2(n)))}$

Part b)

```
for (int i=1; i<=n; i++){
    if ( (i % (int)sqrt(n)) == 0 ){
        for (int k=0; k < pow(i,3); k++){
```

- First line of code does 1...n iterations so it does
- $\theta(n)$
- $\sum_{i=0}^{n} (\theta(1) + O(\sum_{k=0}^{i^3} \theta(1))$
- $\sum_{i=0}^{n} (\theta(1)) + \sum_{i=0}^{\sqrt{n}} i^3 = \theta(n) + \theta(n^{\frac{3}{2}}) = \theta(n^{\frac{3}{2}})$

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)
```

- The first and second for loop which are the outer and middle loop increase by 1 up to n, thus both lines of code each run $\theta(n)$.
- $\sum_{i=1}^{n} \sum_{k=1}^{n} \theta(1) + O(\sum_{m=1}^{n} \theta \log n)$
- $\sum_{i=1}^{n} \sum_{k=1}^{n} \theta(1) + \sum_{k=1}^{n} \sum_{m=1}^{n} \theta \log n = \theta(n) \cdot \theta(n) + \theta(\log n) = \theta(n^2 \cdot \log n)$

- Therefore this whole function runs $\boxed{\theta(n \cdot n \cdot \log(n)) = \theta(n^2 \cdot \log(n))}$

Part d)

- The for loop for(int i=0; i<n; i++) runs n iterations.
- The inner loop, for(int j=0; j < size; j++), size increases by $\frac{3}{2} \cdot size$, ~~thus~~ it goes from 10, 15, 22..., but it is not with respect to n. $10\left(\frac{3}{2}\right)^k \geq n \rightarrow \left(\frac{3}{2}\right)^k \geq \frac{n}{10} \rightarrow k \geq \log_{\frac{3}{2}}\left(\frac{n}{10}\right)$
- Therefore this ~~is~~

$$\sum_{i=0}^{n-1} \theta(1) + \sum_{j=0}^{size} \left(\frac{3}{2}\right)^j = \theta n + \sum_{k=0}^{\log_{\frac{3}{2}} n} \frac{3}{2}^k = \theta(n) + \theta(n)$$

- Runtime $\theta(n)$

Problem 2: Linked List Recursion Tracing

Question a: in1 = 1,2,3,4 in2 = 5,6

Call to function(

Step 1: in1->next = llrec(in2, in1->next)

Step 2: llrec(in2 ([5]), in1->next ([2])) ⇉ in2->next = llrec(in1, in2->next)

Step 3: llrec(in1 [2], in2->next [6]) = in1->next = llrec(in2, in1->next)

Step 4: llrec(in2 [6], in1->next [3]) = in2->next = llrec(in1, in2->next)

Step 5: llrec(in1 [3], in2->next [null]) = return in1.

- Starting at step 5, the value returned is the node [3] from in1.
- Thus in step 4, in2->next, which is the null pointer of node [6], will now point to [3].
- Now in step 3, in1->next, is the pointer in which node [2] will point to, and from step 4, it will point to node [6].
- Now in step 2, in2->next, which is the pointer in which node [5] will point to, and from step 3, it will point to node [2].
- Finally in step 1, since it starts at node 1, then it will point to node [5] because of step 2.

- It returns, 1,5,2,6,3,4

Question b: in1 = nullptr in2 = 2

Step 1: in1 = nullptr, so return in2.

- It will return 2, as the next pointer of in1 will point to starting node of in2, which is 2 and then there are no more nodes.