1 Part 1 Questions

1. $5n^5 + 2n^4 + \log n^6 = O(n^4)$ False
2. $n + 8n^2 + 10n^3 = O(n)$ False
3. $n! + n^2 = O(n \log n)$ False
4. $2^n + n \log n = O(n!)$ True
5. $\frac{1}{n^2} + 10 = O(\frac{1}{n})$ False
6. $\log n + n \log n + \log(\log n) = \Omega(n)$ True
7. $3^n + n! = \Omega(n^n)$ False
8. $\sqrt{n} + n = \Omega(\log n)$ True
9. $10 + 6n = \Omega(1)$ True
10. $n * n^{\frac{3}{2}} = \Omega(n^2)$ True
11. $n^2 + \frac{n}{2} + 1 = \Theta(n^3)$ False

    Since n^3 grows faster than n^2 there is no n_0 and c that will satisfy the equation $f(n) \geq cg(n)$.

12. $\frac{1}{n^{100}} + \log 64 = \Theta(1)$ True

    log 64 = 6 (Base 2)
    for O(1)
    pick c = 7 and $n_0 = 1$
    $\frac{1}{n^{100}} + 6 \leq 7$
    for $\Omega(1)$
    pick c = 1 and $n_0 = 1$
    $\frac{1}{n^{100}} + 6 \geq 1$

13. $n \log n + n = \Theta(\log n)$ False

    Since $n \log n$ grows faster than $\log n$ there is no n_0 and c that will satisfy the equation $f(n) \leq cg(n)$.

14. $2^n + n = \Theta(n)$ False

    Since 2^n grows faster than n there is no n_0 and c that will satisfy the equation $f(n) \leq cg(n)$.

15. $\frac{n^5}{n^2} + \log n = \Theta(n^3)$ True

for O(n^3)
pick c = 2 and $n_0 = 1$
$\frac{n^5}{n^2} + \log n \leq 2n^3$
for $\Omega(1)$
pick c = 1 and $n_0 = 1$
$\frac{n^5}{n^2} + \log n \geq n^3$

2 Part 2

1) num=0;
for (i = n; i>= 0; i−−)
num−−;
First statement = 1 instruction for num
1 instruction for i = n
for every iteration of for, 3 instructions (one for condition check, one for decrement of i and one for decrement of num)
1 instruction for failed condition of for

$T(n) = 3 + \sum_0^n 3$
= 3 + 3n + 3
= O(n)

2) num=0;
for (i = 0; i<= $n * n$; i=i+2)
num=num+2;
First statement = 1 instruction for num
1 instruction for i = 0
for every iteration of for, 3 instructions (one for condition check, one for decrement of i and one for increment of num). There are $\frac{n*n}{2}$ such iterations
1 instruction for failed condition of for

$T(n) = 3 + \sum_0^{\frac{n*n}{2}} 3$
= 3 + 1.5(n*n) + 3
= O(n^2)

3) num=0;
for (i = 1; i<= n; i=i*2)
num++;

First statement = 1 instruction for num
1 instruction for i = 1

for every iteration of for, 3 instructions (one for condition check, one for update of i and 1 for increment of num). There are $\log n$ such iterations
1 instruction for failed condition of for

$T(n) = 3 + \sum_1^{\log n} 3$
$= 3 + 3\log n$
$= O(\log n)$

4) num=1;
for (i = 0; i<n; i++)
for (j = 0; j<=i; j++)
num = num * 2;

First statement = 1 instruction for num
Outer Loop: 1 instruction for i = 0, inner for loop, 1 instruction for increment for i and 1 for condition for i, for n iterations. 1 instruction for failed condition
Inner loop: for every iteration of for, 3 instructions (one for condition check, one for increment of j and one for update of num). There are i such iterations. Moreover, 1 instruction for failed condition of for and 1 for initialization of j.

$T(n) = 3 + \sum_0^{n-1}(4 + \sum_0^i 3)$
$= 3 + \sum_0^{n-1}(4 + 3(i+1))$
$= 3 + 7n + 3(1 + 2 + 3 \ldots n)$ [Ignoring some of the constants]
$= O(\frac{n(n+1)}{2})$
$= O(n^2)$

5) p=10;
num=0;
plimit=100000;
for (i = p; i<=plimit; i++)
for (j = 1; j<=i; j++)
num = num + 1;

3 instructions for p, num and plimit respectively
Outer Loop: 1 instruction for i = p, constant number of loops of inner for, 2 instructions per inner loop, 1 instruction for failed condition of outer loop
Inner Loop: 1 instruction for initialization, 3 instructions per loop, 1 failed condition instruction

$T(n) = 5 + \sum_p^{plimit}(4 + \sum_1^i 3)$
$= 3 + \sum_p^{plimit}(4 + 3i)$
$= 3 + 4(\text{constant}) + 3(\text{constant})$
$= O(\text{constant})$

$= O(1)$

6) num=0;
for (i = n*n; i>0; i=i/2)
for (j = 1; j<=n; j++)
num = num + i;

1 instruction for initialization of num
Outer loop: 1 intruction for i = n*n, 1 for failed condition and for every loop:- updating i, condition for i and inner for loop. Outer loop runs for $\log n^2$ time.
Inner loop: 1 for initialization of of j, 1 for failed condition, for every loop:- updating j, condition for j and updating num
$T(n) = 3 + \sum_0^{2\log n}(4 + \sum_1^n 3)$
$= O(3 + 8\log n + 3n\log n)$
$= O(n\log n)$

7) num=0;
for (i = 0; i<n*n-1; i++)
for (j = 0; j< $i * i$; j++)
num = 2*num;

1 instruction for initialization of num
Outer loop: 1 intruction for initialization of i, 1 for failed condition and for every loop:- updating i, condition for i and inner for loop. Outer loop runs for $n^2 - 1$ times.
Inner loop: 1 for initialization of of j, 1 for failed condition, for every loop:- updating j, condition for j and updating num. Inner loop runs i^2 times.
$T(n) = 3 + \sum_0^{n^2-2}(4 + \sum_0^{i^2-1} 3)$
$= O(3 + 4n^2 + \sum_0^{n^2-2} 3i^2)$ [Ignoring some of the constants]
$= O(3 + 4n^2 + 3(1^2 + 2^2 + 3^2 + ... + (n^2)^2))$
$= O(3 + 4n^2 + 3 * \frac{n^2*(n^2+1)*(2n^2+1)}{6})$
$= O(constant + n^2 + n^4 + n^6)$
$= O(n^6)$

8) num=0;
for (i = 0; i<= n*n; i++)
num++;
for (i = 1; i<=n; i=i*2)
for (j=n; j>= 1; j=j/2)
num++;

First for loop: 1 instruction for initialization, 1 for failed condition, for every loop:- 1 condition, 1 increment and one for num. Loop runs $n^2 + 1$ times.
Second for loop:

Outer loop: 1 instruction for initialization of i, 1 for failed condition, for every loop:- 1 condition, 1 increment and one inner loop. Outer loop runs $\log n$ times.

Inner loop: 1 initialization, 1 failed condition, for every loop:- 1 condition, 1 increment and 1 for num. Loop runs $\log n$ times.

$$\begin{aligned}
\text{T(n)} &= 3 + \sum_0^{n^2} 3 + 2 + \sum_1^{\log n}(3 + \sum_1^{\log n} 3) \\
&= \text{O(constant} + n^2 + \log n + \log n * \log n) \\
&= \text{O}(n^2)
\end{aligned}$$

```
9) for (i = 0; i < n; i++) {
        smallest = i;
        for (j = i+1; j <= n; j++) {
              if (a[j] < a[smallest])
                    smallest = j;
        }
        swap(a, i, smallest); // has three instructions
}
```

Outer loop: 1 instruction for initialization of i, 1 for failed condition, for every loop:- 1 condition, 1 increment, 1 for update of smallest, inner loop, 3 for swap operation. Loop runs n times.
Inner loop: 1 instruction for initialization of, 1 for failed condition, for every loop:- 1 condition, 1 increment, 2 for the if in worst case. Loop runs $n - i$ times.

$$\begin{aligned}
T(n) &= 2 + \sum_0^{n-1}(8 + \sum_{i+1}^{n} 4) \\
&= 2 + 8\text{n} + \sum_0^{n-1} 4(n - i) \\
&= \text{O(constant} + \text{n} + (n + (n-1) + (n-2) + ... + 2 + 1)) \\
&= \text{O(constant} + \text{n} + \tfrac{n*(n+1)}{2}) \\
&= \text{O}(n^2)
\end{aligned}$$

```
10) num = 0;
i = 0;
while (i<n) {
        j = 0;
        while (j<100) {
              //constant time operations
              j++;
        }
        i++;
}
```

2 instructions for num and i initialization
Outer while loop: 1 instruction for failed condition, for each loop:- 1 for update of j, 1 for increment of i and inner loop. This loop runs n times.

Inner while loop: 1 instruction for failed condition, for each loop:- 1 for condition check, 1 for update of j, constant number of instructions marked by comment. This loop runs 99 (or constant) times.

$T(n) = 3 + \sum_0^{n-1}(4 + \sum_0^{99} X)$
$= \text{O (constant + n)}$
$= \text{O(n)}$

3 Part 3

1. Copying the list.
   Singly Linked List: Copying the list involves going through all the nodes one by one. Takes $\theta(n)$ time.
   ArrayList: Copying an array based list involves accessing each index once. Takes $\theta(n)$ time.

2. Adding a value to the middle of the list. (where you do not want to destroy the overall order of the elements)
   Linked List: We would need to move through $\frac{n}{2}$ nodes in the list assuming we already know n to add a node to the middle of the list. This operation takes $\theta(n)$ time.
   ArrayList: Accessing the middle element in the array would take constant time. However, we will have to copy the second half of the array and move it by one position to the right. This operation takes $\theta(n)$ time. The overall complexity is $\theta(n)$.

3. Removing the value from the list at a given index. (where you do not want to destroy the order of the elements).
   Linked List: We will have to move to the node at the given index which involves traversing through the entire list in the worst case. This operation takes $\theta(n)$.
   Array List: Access the index take constant time. However, we will have to copy the second half of the array and move it by one position to the left. This operation takes $\theta(n)$ time. The overall complexity is $\theta(n)$.

4. Removing the first value from the list.
   Linked List: Removing the first element involves only modifying the head node which takes $\theta(1)$ time.
   Array List: Removing the first element requires the entire list to be copied by one position to the left. This operation take $\theta(n)$ time.

5. Determining whether the list contains some value v.
   Linked List: The search operation requires us to traverse through the entire list in the worst case which takes $\theta(n)$ time.
   Array List: Since the ArrayList is not sorted, the search operation requires us to traverse through the entire array which takes $\theta(n)$ time.

4 Part 4

Plot for Linked List should be linear - $\theta(n)$ as the RemoveAllFront removes the element at the front(which takes constant time) for n elements.
Plot for array should be quadratic - $\theta(n^2)$ as the RemoveAllFront removes the element at the front (which takes linear time) for n such elements.