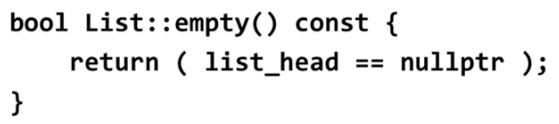
1. **Onsay**: For the following program segment, give an analysis of the running time for Big O.

```
Sum=0
For i=1 to $N^2$ do
    For j=1 to N do
        Sum= Sum +1
```

a. O(N) b. O(N^2) c. O(N^3) d. O(2^N)

Answer: C First loop invariant (the loop counter)grows quadratically cost time analysis is : (n^2+1) The inner loop (2nd loop) cost time$\sum_{i=1}^{n^2}((t_i + 1)^2$, sum notation cost time analysis is n. When solving for T(n) you will get O(n^3)

2. **Onsay**: Below is a Boolean value returning function named as empty, to check if the linked list is empty. What would be the Big O run time analysis of this following algorithm.



```
bool List::empty() const {
    return ( list_head == nullptr );
}
```

a. O(log n) b. O(nlogn) c. O(n) d. O(1)

Answer d

3. **Onsay**: Based on Big $\theta, O \text{ and } \Omega$ definition. Determine whether the following assertion is true or false

$$\frac{n(n+1)}{2} \in \theta(n^3)$$

a. True b. False

Answer False. n^2 is not $\theta(n^3)$

4. **Onsay**: For the following pair of functions, indicate whether the **first** function has a lower, same, or higher order of growth (to within a constant multiple) than the **second** function.

First function: 2^{n-1} Second function: 2^n

a. Lower order of growth b. Same order of growth c. Higher order of growth

Answer: B (In first function -1 is a constant, that will not change the growth magnitude and will grow almost at the same rate as $2^{n)}$

5. **Onsay**: Please see the algorithm below, what does this algorithm do and what would be its Big θ?

```
MysteryAlgorithm(A[0..n-1])
//Input: An array A[0..n-1]
//output boolean
for i ← $0\ to\ n-2\ do$
    for $j \leftarrow i+1\ to\ n-1\ do$
        if $A[i] = A[j]\ return\ false$
$return\ true$
```

a. Returns false if all elements in A are swapped and Big θ is $\theta(1)$

b. Returns true if all elements are sorted in ascending order and Big θ is $\theta(n^2)$

c. Returns true after counting the occurrences of an entry and Big θ is $\theta(n)$

d. Returns true if all elements in A are distinct and Big θ is $\theta(n^2)$

Answer: D

6. **Onsay**: Below is the algorithm for Bubble Sort.

```
BubbleSort(A[0..n-1])
//Input: An Array A[0..n-1]
//Output: An array A[0..n-1] sorted in an increasing order
for i ← 0 to n - 2 do
    for j ← 0 to n – 2 - i do
        if A[j+1] < A[j] swap A[j] and A[j+1]
```

Array A contains the following values and **no iterations** are done at this point.

99	**45**	**68**	**90**	**29**	**34**	**17**	**values stored in array A,**
[0]	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**index values of array A**

What would array A look like after completing <u>**3 iterations of the outer for loop**</u>?

a.	45	68	90	29	34	17	99
b.	45	68	29	34	17	90	99
c.	45	29	34	17	68	90	99
d.	45	68	90	99	29	34	17

Answer: c

7. **Tito**: Stacks should not be used to :

a) evaluation of expression in postfix form
b) implementation of recursion
c) reversing string
d) allocating resources and scheduling
e) all of the above

Answer: D Because as we know that stack is a LIFO data structure. If it is used for scheduling, then the initial processes will not get any CPU time till the stack gets cleared above them and this will lead to starvation.)

8. **Tito**: Evaluate potsfix expression from given infix expression.

A + B * (C + D) / F + D * E

A. ABCD+*F/+DE*+
B. AB+CD*F/+DE*
C. AB+CD*F/+D*E
D. ABCD+*/F+DE*
E. AB+CD+*F/DE+*+

Answer: A (Note that C+D is evaluated first followed by the multiplication with B. Hence ABCD come together, followed by the division with F)

9. **Wei**:

```
i=1
while(i<=n)
    i=i*3
```

Please select one of the choice for the time complexity in T(n) .

a) O(3log$_2$n) b) O(nlog$_3$n) c) O(log$_3$n) d)O(log$_3$log$_3$(n))

Answer: cost time analysis:

i=1 → 1
while(i<=n) → log$_3$n+1
i=i*3 → log$_3$n
Solving for T(n)
T(n)=2log$_3$n+2
T(n)=O(log$_3$n)

10. **Wei:** Use binary search to find the biggest item in an array with 64 items in order, how many times of search are needed at most?

a) 5 b) 6 c) 7 d) 1
Answer: b → 6 times.
Worst case, key is the first item. Then check 32, 16, 8, 4, 2, 1 position.

11. **Scott:** What operation does the Insertion Sort use **to move numbers from** the unsorted section to the sorted section of the list?

A. Finding the minimum value
B. Swapping
C. Finding out a pivot value
D. None of the above

Answer : B Once a smaller value is found, the old value at that index is swapped out for the new value.

12. **Scott**: If all the elements in an input array is equal for example {1,1,1,1,1,1}, What would be the running time of the Insertion Algorithm?

A.
O(2^N)
B. O(n^2)
C. O(n)
D. None of the above

Answer: C The best case input is an array that is already sorted. In this case insertion sort has a linear running time (i.e., O(n)). During each iteration, the first remaining element of the input is only compared with the right-most element of the sorted subsection of the array.