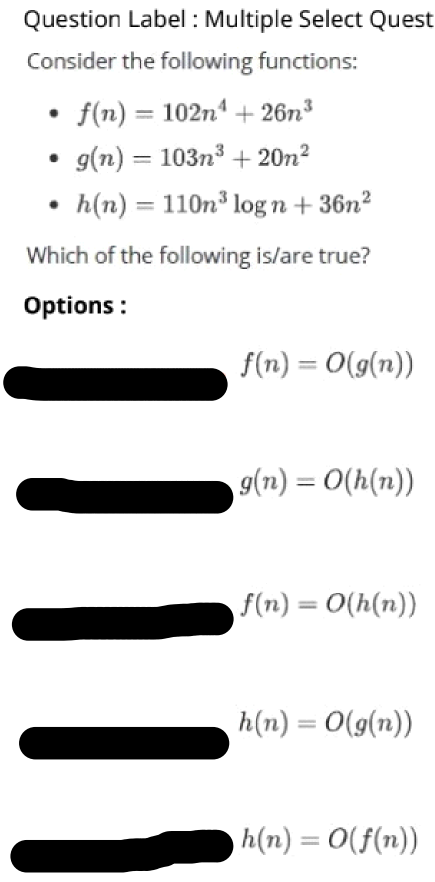
Week 2: PYQs

06 June 2024 09:56

1. Jan 24, Quiz 1



Question Label : Multiple Select Question

Consider the following functions:

- $f(n) = 102n^4 + 26n^3$
- $g(n) = 103n^3 + 20n^2$
- $h(n) = 110n^3 \log n + 36n^2$

Which of the following is/are true?

Options :

[redacted] $f(n) = O(g(n))$

[redacted] $g(n) = O(h(n))$

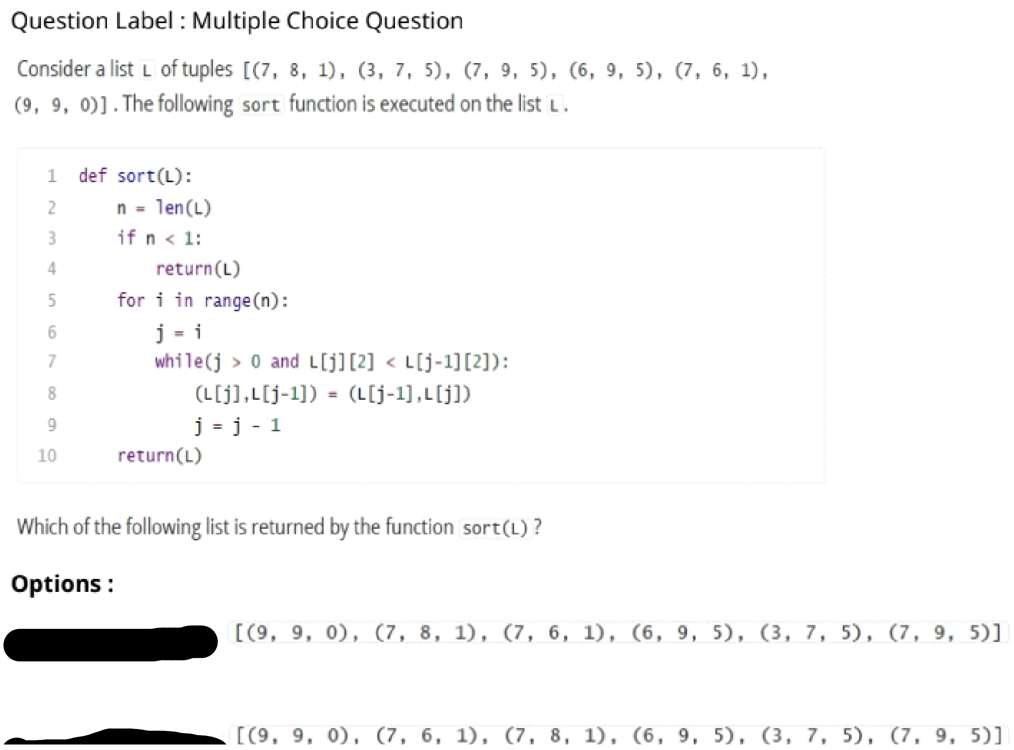
[redacted] $f(n) = O(h(n))$

[redacted] $h(n) = O(g(n))$

[redacted] $h(n) = O(f(n))$

Screen clipping taken: 06-06-2024 10:08

2. Sep 22, Quiz 1



Question Label : Multiple Choice Question

Consider a list `L` of tuples `[(7, 8, 1), (3, 7, 5), (7, 9, 5), (6, 9, 5), (7, 6, 1), (9, 9, 0)]`. The following `sort` function is executed on the list `L`.

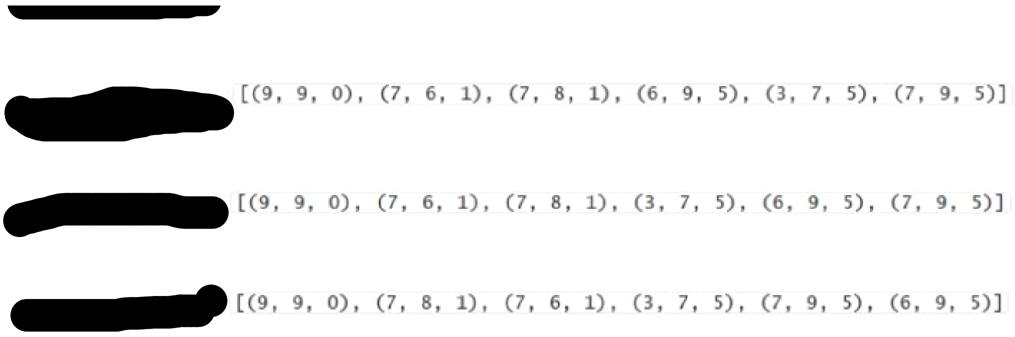
```
def sort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        j = i
        while(j > 0 and L[j][2] < L[j-1][2]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j - 1
    return(L)
```

Which of the following list is returned by the function `sort(L)`?

Options :

[redacted] `[(9, 9, 0), (7, 8, 1), (7, 6, 1), (6, 9, 5), (3, 7, 5), (7, 9, 5)]`

[redacted] `[(9, 9, 0), (7, 6, 1), (7, 8, 1), (6, 9, 5), (3, 7, 5), (7, 9, 5)]`



[redacted]

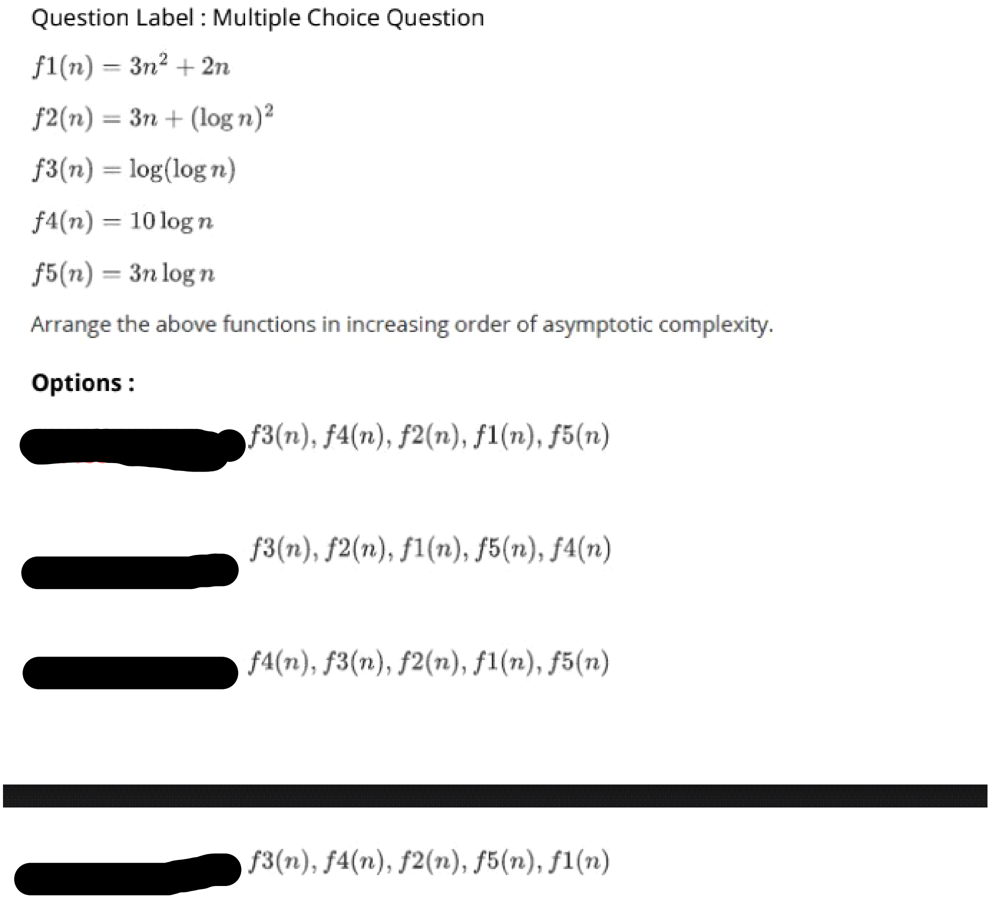
[redacted] `[(9, 9, 0), (7, 6, 1), (7, 8, 1), (6, 9, 5), (3, 7, 5), (7, 9, 5)]`

[redacted] `[(9, 9, 0), (7, 6, 1), (7, 8, 1), (3, 7, 5), (6, 9, 5), (7, 9, 5)]`

[redacted] `[(9, 9, 0), (7, 8, 1), (7, 6, 1), (3, 7, 5), (7, 9, 5), (6, 9, 5)]`

Screen clipping taken: 06-06-2024 09:57

3. Jan 2023, Quiz 1



Question Label : Multiple Choice Question

$f1(n) = 3n^2 + 2n$

$f2(n) = 3n + (\log n)^2$

$f3(n) = \log(\log n)$

$f4(n) = 10 \log n$

$f5(n) = 3n \log n$

Arrange the above functions in increasing order of asymptotic complexity.

Options :

[redacted] $f3(n), f4(n), f2(n), f1(n), f5(n)$

[redacted] $f3(n), f2(n), f1(n), f5(n), f4(n)$

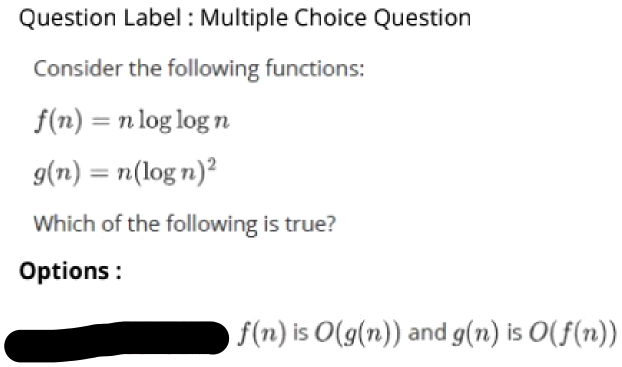
[redacted] $f4(n), f3(n), f2(n), f1(n), f5(n)$

[redacted]

[redacted] $f3(n), f4(n), f2(n), f5(n), f1(n)$

Screen clipping taken: 06-06-2024 10:03

4. Sep 23, Quiz 1



Question Label : Multiple Choice Question

Consider the following functions:

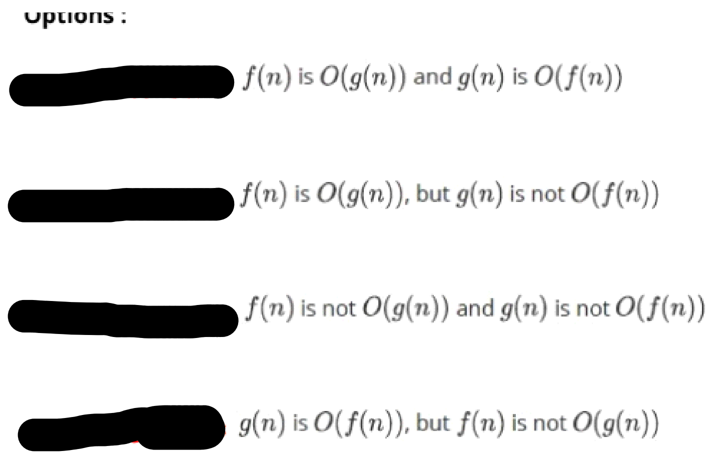
$f(n) = n \log \log n$

$g(n) = n(\log n)^2$

Which of the following is true?

Options :

[redacted] $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$



Options :

[redacted] $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$

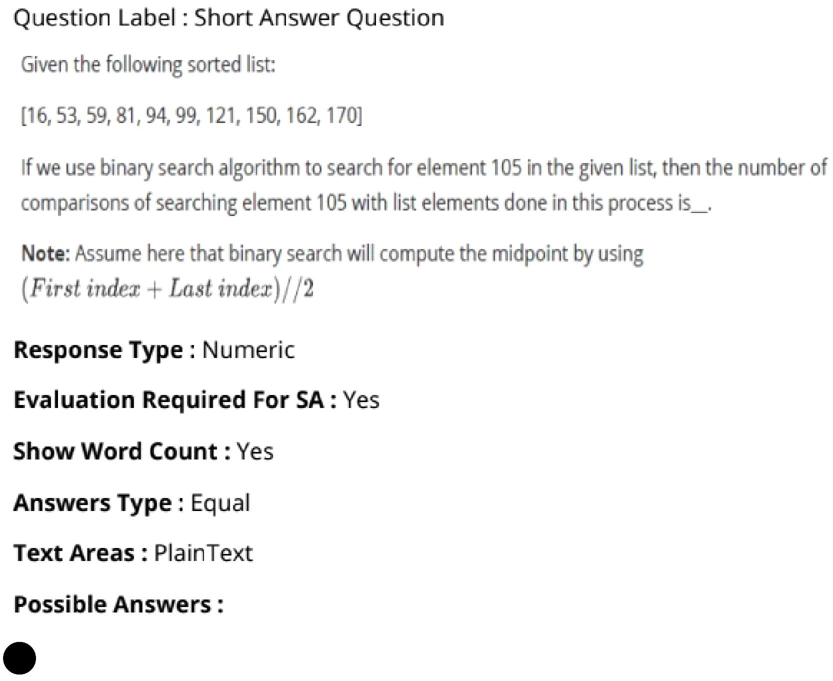
[redacted] $f(n)$ is $O(g(n))$, but $g(n)$ is not $O(f(n))$

[redacted] $f(n)$ is not $O(g(n))$ and $g(n)$ is not $O(f(n))$

[redacted] $g(n)$ is $O(f(n))$, but $f(n)$ is not $O(g(n))$

Screen clipping taken: 07-06-2024 15:52

5. Sep 23, Quiz 1



Question Label : Short Answer Question

Given the following sorted list:

[16, 53, 59, 81, 94, 99, 121, 150, 162, 170]

If we use binary search algorithm to search for element 105 in the given list, then the number of comparisons of searching element 105 with list elements done in this process is__.

Note: Assume here that binary search will compute the midpoint by using $(First\ index + Last\ index)//2$

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

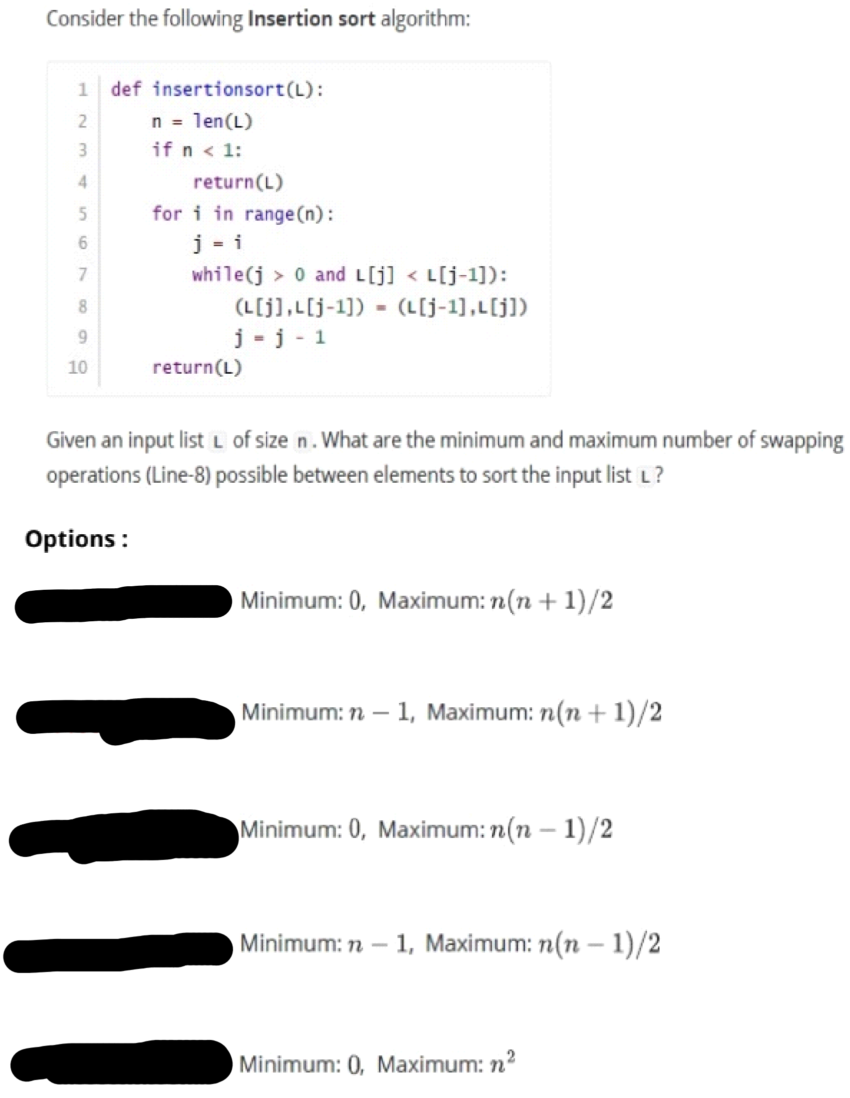
Text Areas : PlainText

Possible Answers :

[redacted]

Screen clipping taken: 07-06-2024 15:53

6. Sep 23, Quiz 1



Consider the following **Insertion sort** algorithm:

```
 1  def insertionsort(L):
 2      n = len(L)
 3      if n < 1:
 4          return(L)
 5      for i in range(n):
 6          j = i
 7          while(j > 0 and L[j] < L[j-1]):
 8              (L[j],L[j-1]) = (L[j-1],L[j])
 9              j = j - 1
10      return(L)
```

Given an input list `L` of size `n`. What are the minimum and maximum number of swapping operations (Line-8) possible between elements to sort the input list `L`?

Options :

[redacted] Minimum: 0, Maximum: $n(n+1)/2$

[redacted] Minimum: $n-1$, Maximum: $n(n+1)/2$

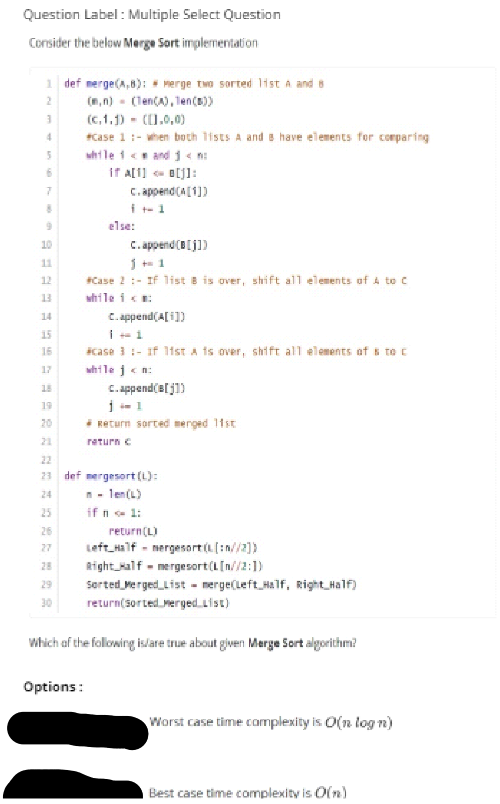
[redacted] Minimum: 0, Maximum: $n(n-1)/2$

[redacted] Minimum: $n-1$, Maximum: $n(n-1)/2$

[redacted] Minimum: 0, Maximum: n^2

Screen clipping taken: 07-06-2024 15:54

7. Jan 2024, Quiz 1



Question Label : Multiple Select Question

Consider the below **Merge Sort** implementation

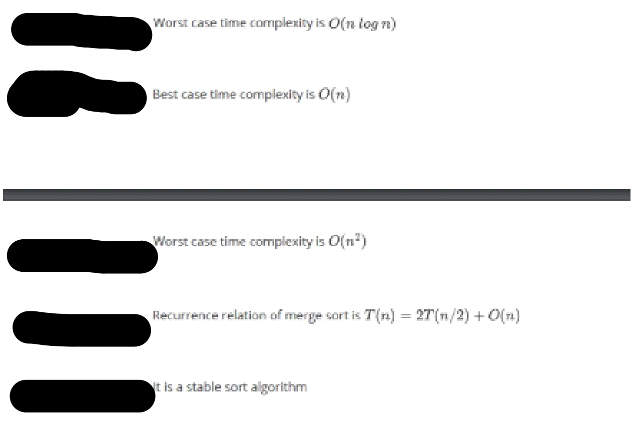
```
def merge(A,B): # Merge two sorted list A and B
    (m,n) = (len(A),len(B))
    (c,i,j) = ([],0,0)
    #Case 1 :- when both lists A and B have elements for comparing
    while i < m and j < n:
        if A[i] <= B[j]:
            C.append(A[i])
            i += 1
        else:
            C.append(B[j])
            j += 1
    #Case 2 :- If list B is over, shift all elements of A to C
    while i < m:
        C.append(A[i])
        i += 1
    #Case 3 :- If list A is over, shift all elements of B to C
    while j < n:
        C.append(B[j])
        j += 1
    # Return sorted merged list
    return C

def mergesort(L):
    n = len(L)
    if n <= 1:
        return(L)
    Left_Half = mergesort(L[:n//2])
    Right_Half = mergesort(L[n//2:])
    Sorted_Merged_List = merge(Left_Half, Right_Half)
    return(Sorted_Merged_List)
```

Which of the following is/are true about given **Merge Sort** algorithm?

Options :

[redacted] Worst case time complexity is $O(n\ log\ n)$

[redacted] Best case time complexity is $O(n)$



[redacted] Worst case time complexity is $O(n \log n)$

[redacted] Best case time complexity is $O(n)$

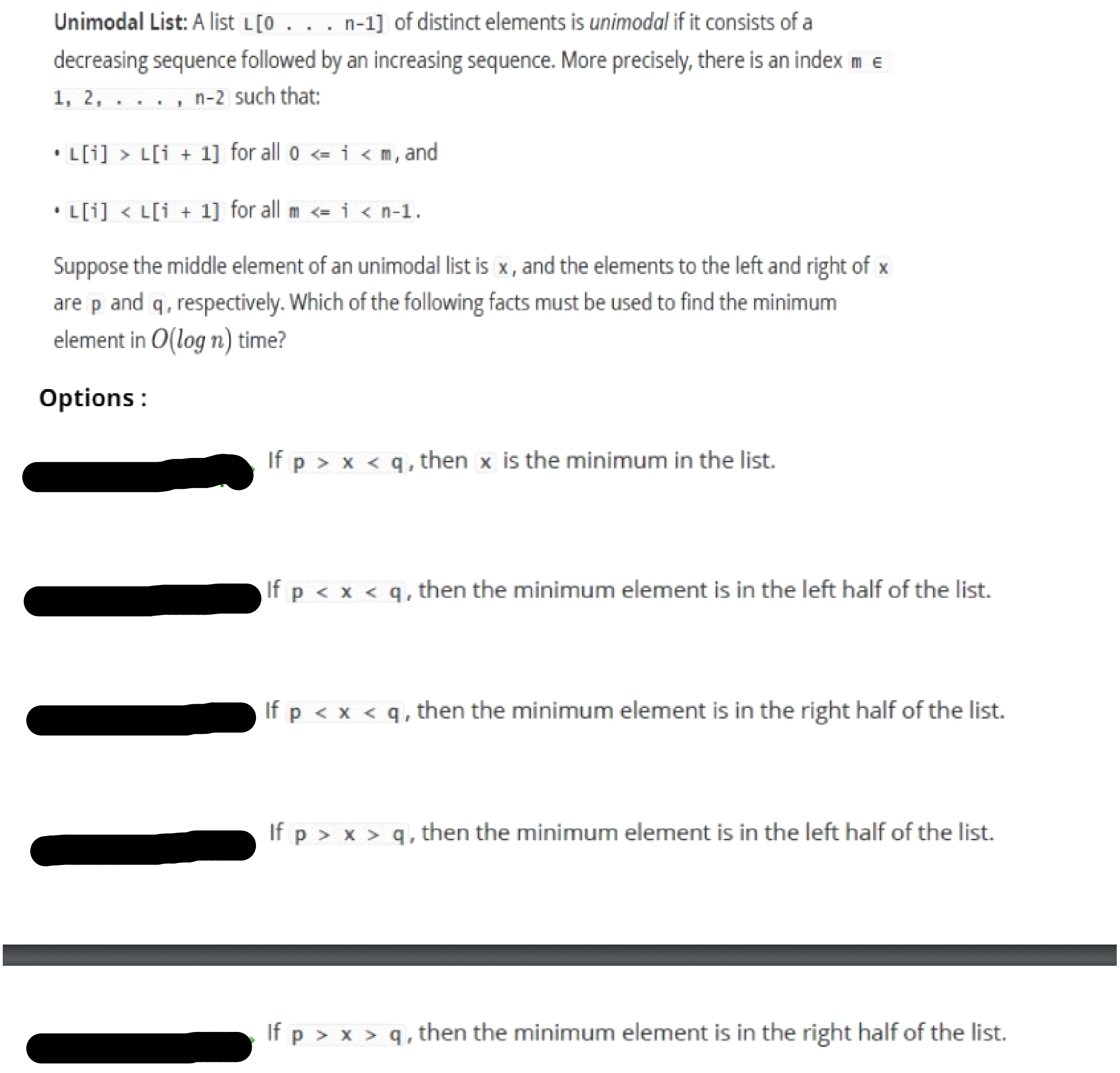
[redacted] Worst case time complexity is $O(n^2)$

[redacted] Recurrence relation of merge sort is $T(n) = 2T(n/2) + O(n)$

[redacted] It is a stable sort algorithm

Screen clipping taken: 14-06-2024 13:10

8. Jan 2024, Quiz 1



Unimodal List: A list `L[0 . . . n-1]` of distinct elements is *unimodal* if it consists of a decreasing sequence followed by an increasing sequence. More precisely, there is an index `m ∈ 1, 2, . . . , n-2` such that:

- `L[i] > L[i + 1]` for all `0 <= i < m`, and
- `L[i] < L[i + 1]` for all `m <= i < n-1`.

Suppose the middle element of an unimodal list is `x`, and the elements to the left and right of `x` are `p` and `q`, respectively. Which of the following facts must be used to find the minimum element in $O(log\, n)$ time?

Options :

[redacted] If `p > x < q`, then `x` is the minimum in the list.

[redacted] If `p < x < q`, then the minimum element is in the left half of the list.

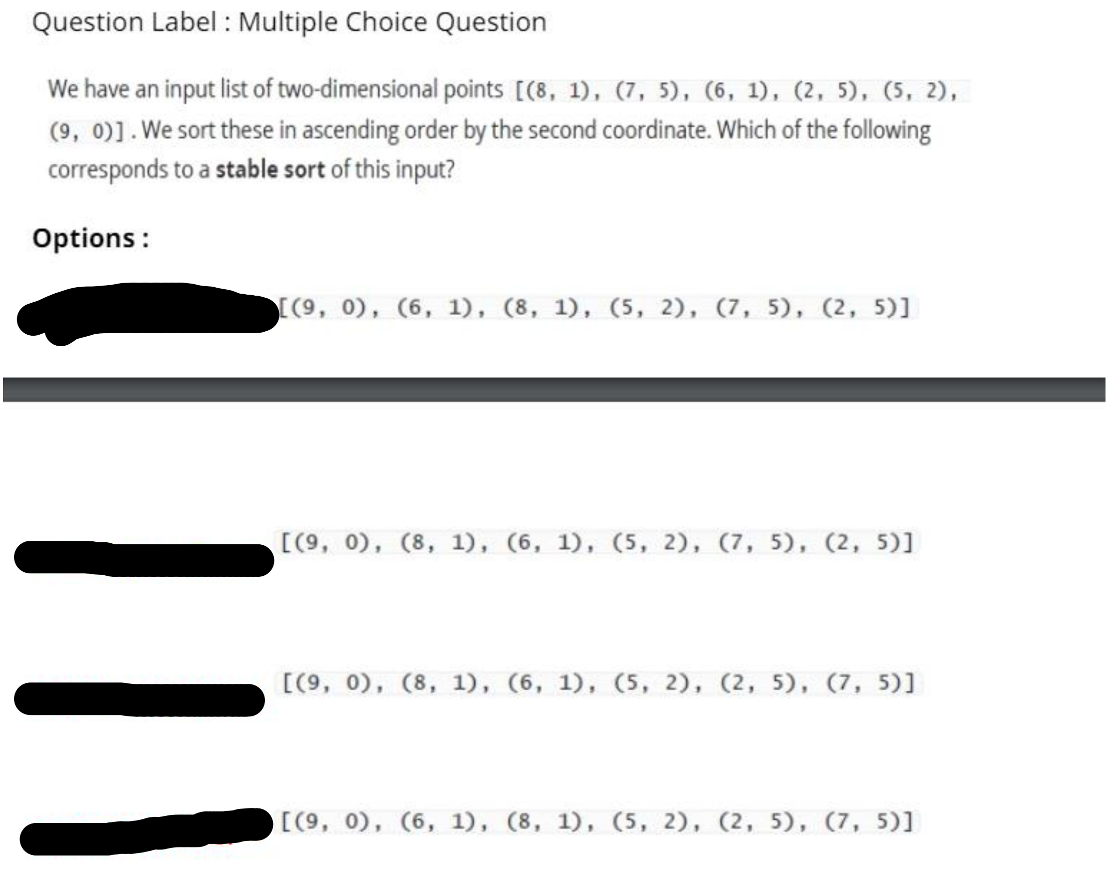
[redacted] If `p < x < q`, then the minimum element is in the right half of the list.

[redacted] If `p > x > q`, then the minimum element is in the left half of the list.

[redacted] If `p > x > q`, then the minimum element is in the right half of the list.

Screen clipping taken: 14-06-2024 13:11

9. Jan 2024, Quiz 1



Question Label : Multiple Choice Question

We have an input list of two-dimensional points `[(8, 1), (7, 5), (6, 1), (2, 5), (5, 2), (9, 0)]`. We sort these in ascending order by the second coordinate. Which of the following corresponds to a **stable sort** of this input?

Options :

[redacted] `[(9, 0), (6, 1), (8, 1), (5, 2), (7, 5), (2, 5)]`

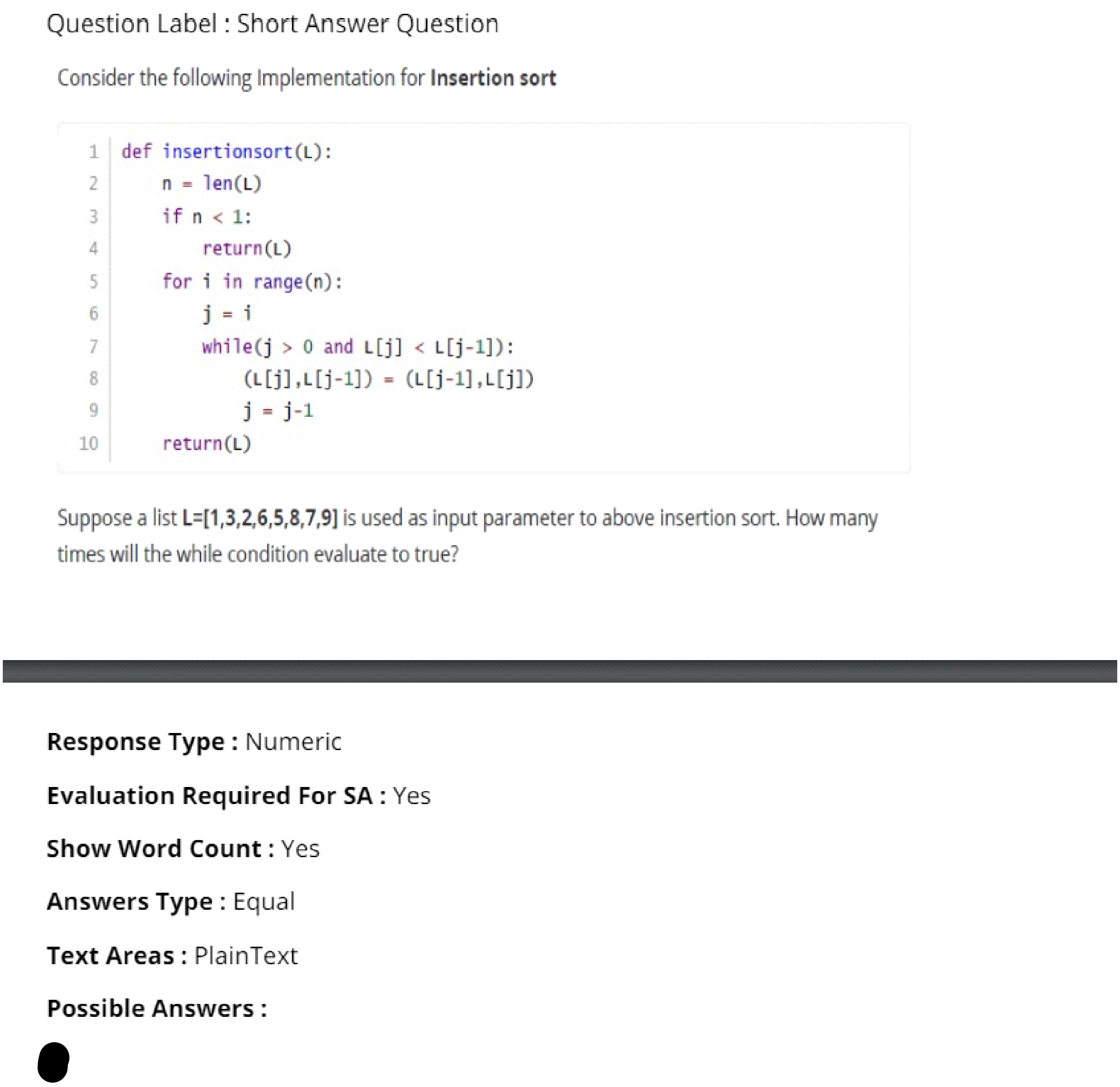
[redacted] `[(9, 0), (8, 1), (6, 1), (5, 2), (7, 5), (2, 5)]`

[redacted] `[(9, 0), (8, 1), (6, 1), (5, 2), (2, 5), (7, 5)]`

[redacted] `[(9, 0), (6, 1), (8, 1), (5, 2), (2, 5), (7, 5)]`

Screen clipping taken: 14-06-2024 13:12

10. Jan 2024, Quiz 1



Question Label : Short Answer Question

Consider the following Implementation for **Insertion sort**

```
def insertionsort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        j = i
        while(j > 0 and L[j] < L[j-1]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j-1
    return(L)
```

Suppose a list **L=[1,3,2,6,5,8,7,9]** is used as input parameter to above insertion sort. How many times will the while condition evaluate to true?

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

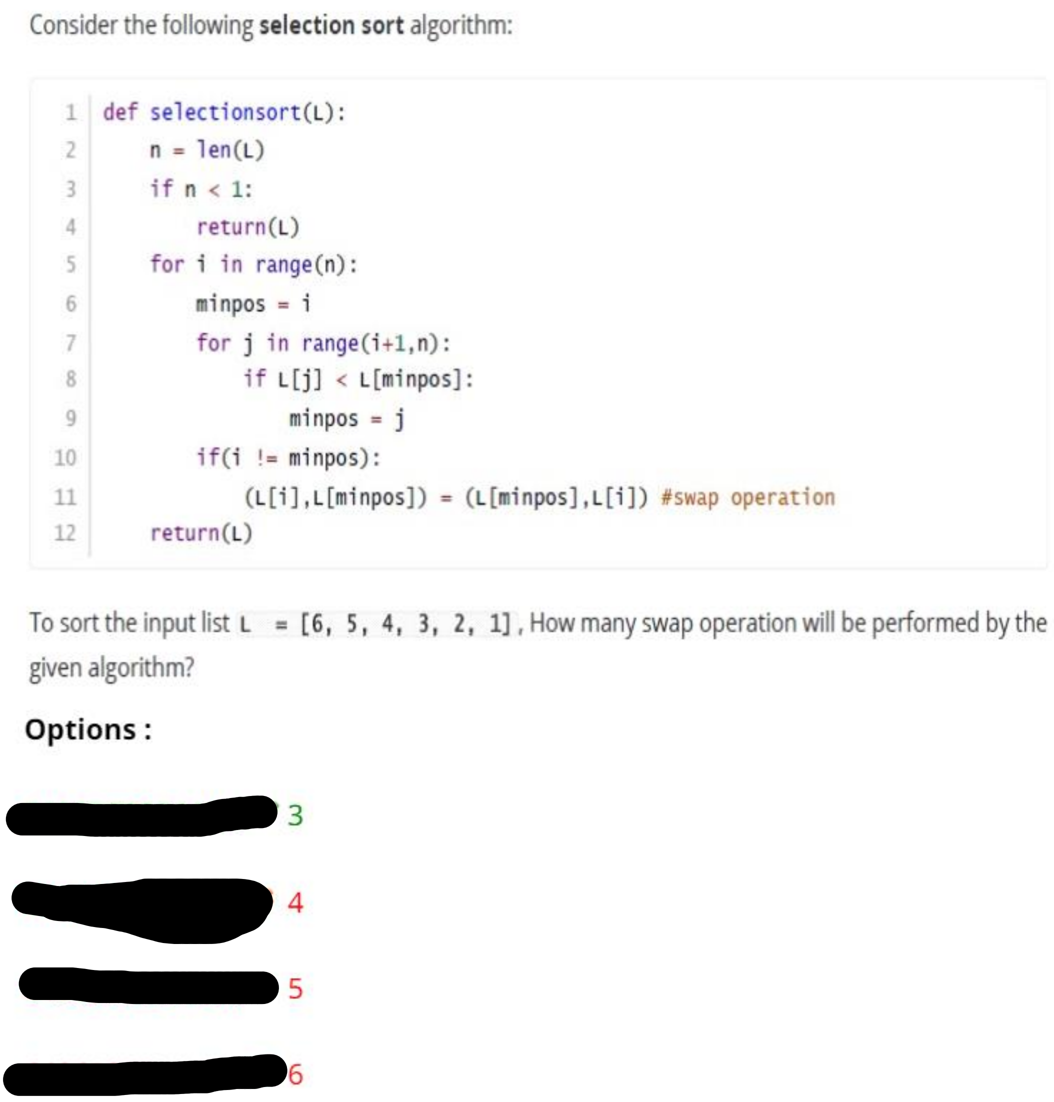
Text Areas : PlainText

Possible Answers :

[redacted]

Screen clipping taken: 14-06-2024 13:13

11. Sep 23, End Term



Consider the following **selection sort** algorithm:

```
def selectionsort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        minpos = i
        for j in range(i+1,n):
            if L[j] < L[minpos]:
                minpos = j
        if(i != minpos):
            (L[i],L[minpos]) = (L[minpos],L[i]) #swap operation
    return(L)
```

To sort the input list `L  = [6, 5, 4, 3, 2, 1]`, How many swap operation will be performed by the given algorithm?

Options :

[redacted] 3

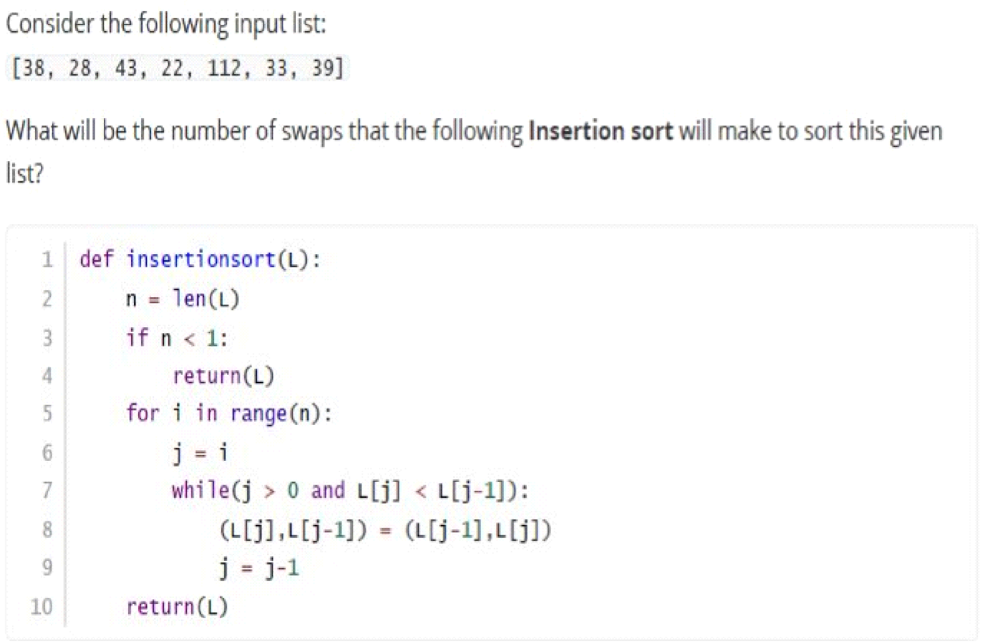
[redacted] 4

[redacted] 5

[redacted] 6

Screen clipping taken: 14-06-2024 13:16

12. Jan 23, Quiz 1



Consider the following input list:

`[38, 28, 43, 22, 112, 33, 39]`

What will be the number of swaps that the following **Insertion sort** will make to sort this given list?

```
def insertionsort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        j = i
        while(j > 0 and L[j] < L[j-1]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j-1
    return(L)
```

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

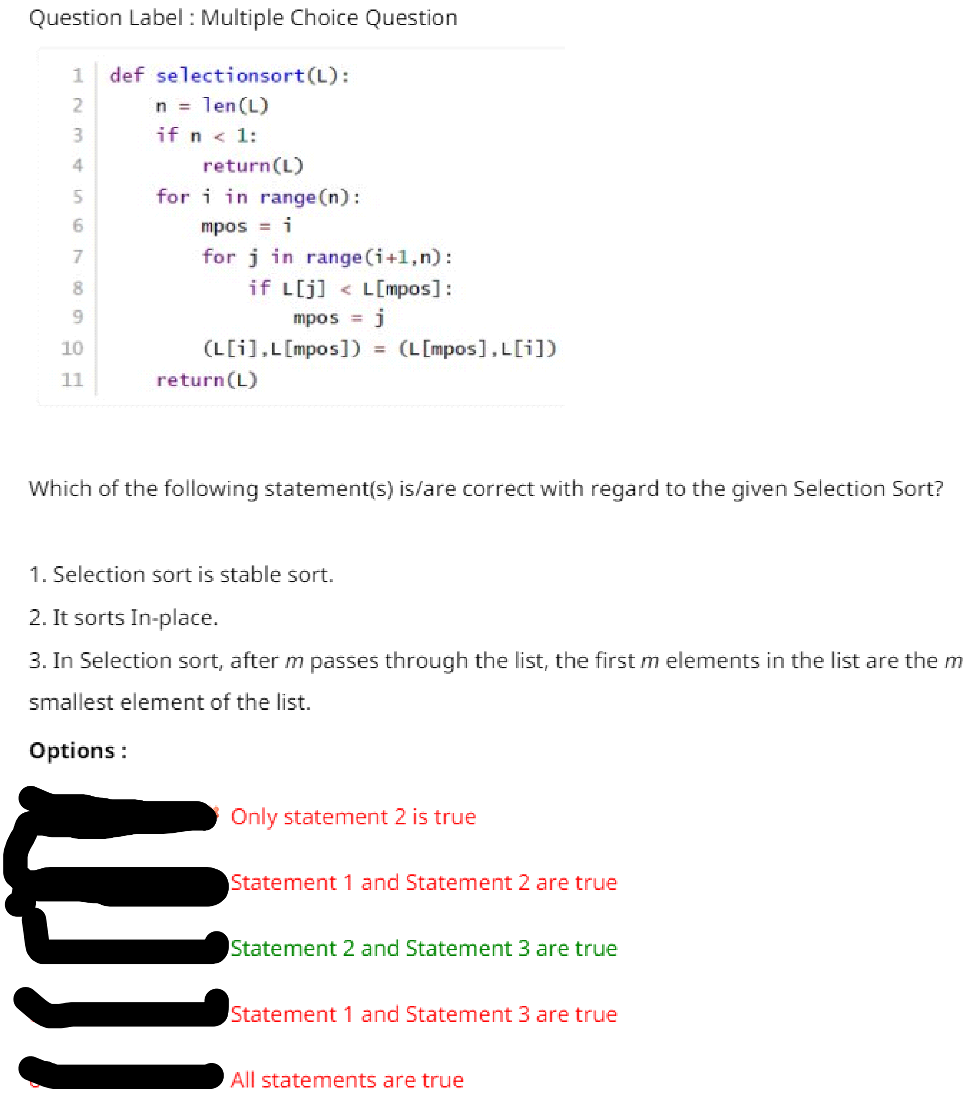
Text Areas : PlainText

Possible Answers :

[redacted]

Screen clipping taken: 14-06-2024 13:32

13. May 23, End Term



Question Label : Multiple Choice Question

```
def selectionsort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        mpos = i
        for j in range(i+1,n):
            if L[j] < L[mpos]:
                mpos = j
        (L[i],L[mpos]) = (L[mpos],L[i])
    return(L)
```

Which of the following statement(s) is/are correct with regard to the given Selection Sort?

1. Selection sort is stable sort.

2. It sorts In-place.

3. In Selection sort, after *m* passes through the list, the first *m* elements in the list are the *m* smallest element of the list.

Options :

[redacted] Only statement 2 is true

[redacted] Statement 1 and Statement 2 are true

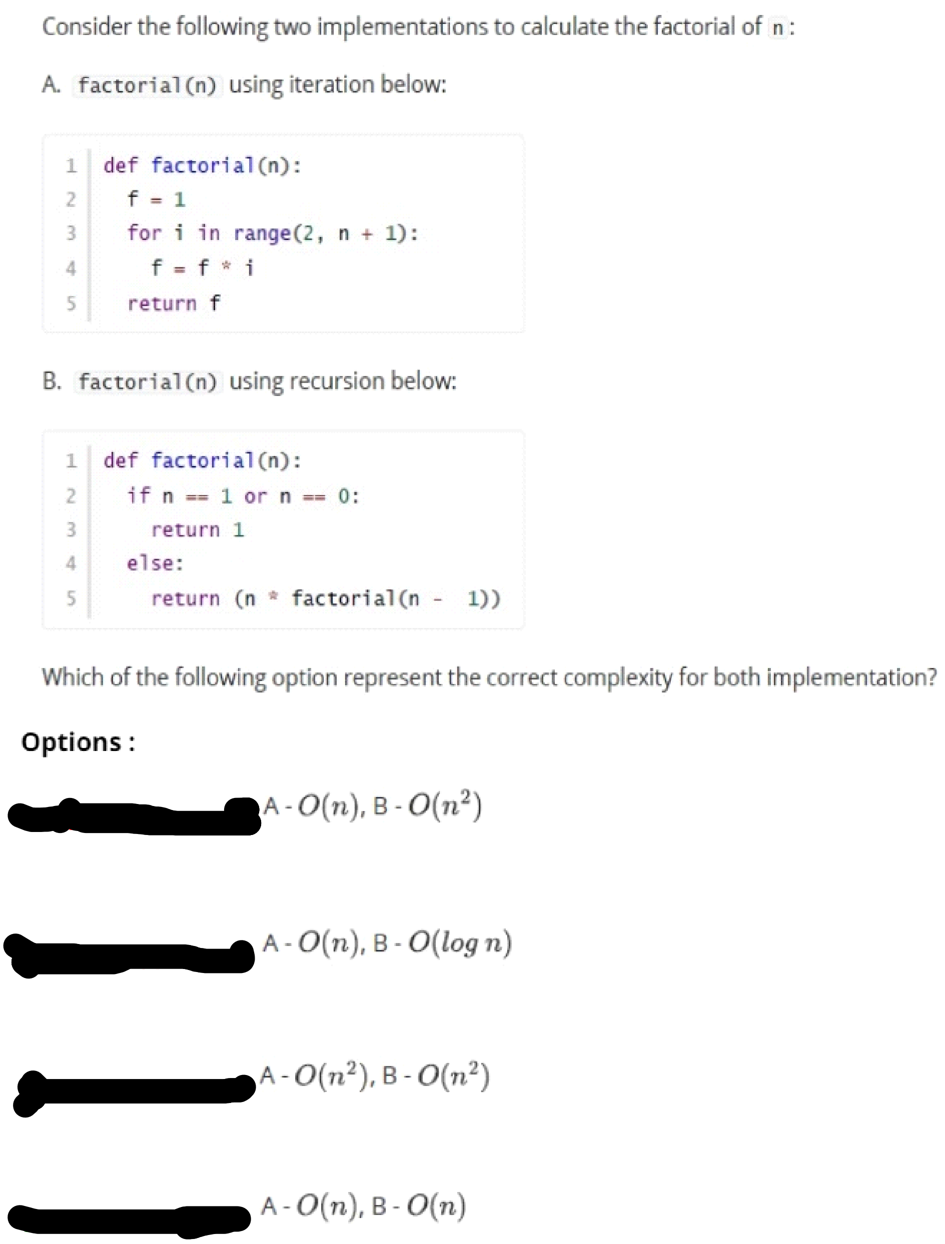
[redacted] Statement 2 and Statement 3 are true

[redacted] Statement 1 and Statement 3 are true

[redacted] All statements are true

Screen clipping taken: 14-06-2024 13:18

14. Sep 23, Quiz 1



Consider the following two implementations to calculate the factorial of `n`:

A. `factorial(n)` using iteration below:

```
def factorial(n):
  f = 1
  for i in range(2, n + 1):
    f = f * i
  return f
```

B. `factorial(n)` using recursion below:

```
def factorial(n):
  if n == 1 or n == 0:
    return 1
  else:
    return (n * factorial(n -  1))
```

Which of the following option represent the correct complexity for both implementation?

Options :

[redacted] A - $O(n)$, B - $O(n^2)$

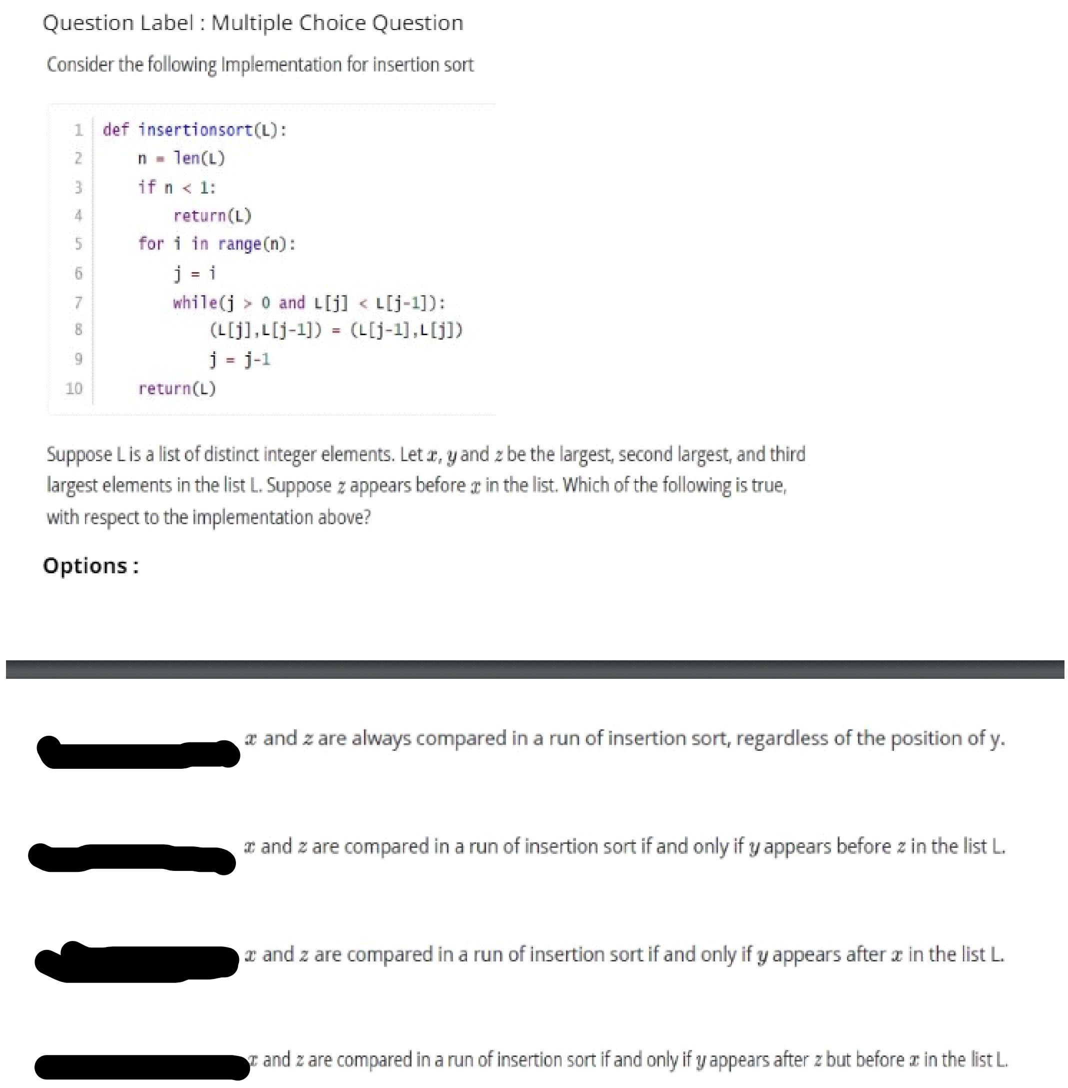
[redacted] A - $O(n)$, B - $O(log\, n)$

[redacted] A - $O(n^2)$, B - $O(n^2)$

[redacted] A - $O(n)$, B - $O(n)$

Screen clipping taken: 14-06-2024 13:23

15. May 23, Quiz 1



Question Label : Multiple Choice Question

Consider the following Implementation for insertion sort

```
def insertionsort(L):
    n = len(L)
    if n < 1:
        return(L)
    for i in range(n):
        j = i
        while(j > 0 and L[j] < L[j-1]):
            (L[j],L[j-1]) = (L[j-1],L[j])
            j = j-1
    return(L)
```

Suppose L is a list of distinct integer elements. Let x, y and z be the largest, second largest, and third largest elements in the list L. Suppose z appears before x in the list. Which of the following is true, with respect to the implementation above?

Options :

[redacted] x and z are always compared in a run of insertion sort, regardless of the position of y.

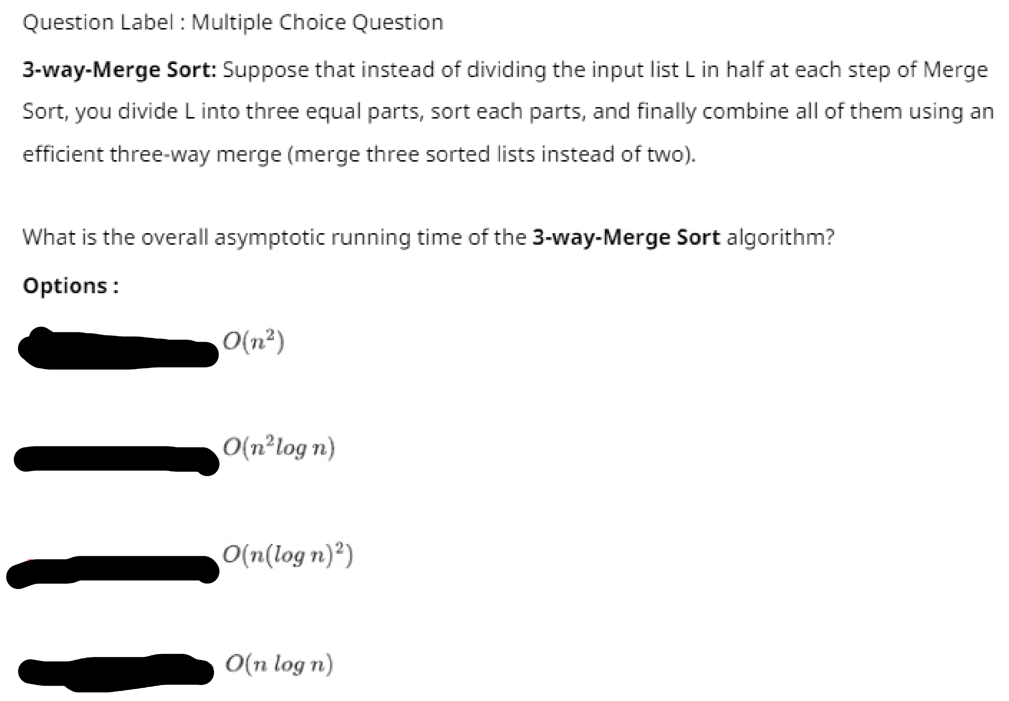
[redacted] x and z are compared in a run of insertion sort if and only if y appears before z in the list L.

[redacted] x and z are compared in a run of insertion sort if and only if y appears after x in the list L.

[redacted] x and z are compared in a run of insertion sort if and only if y appears after z but before x in the list L.

Screen clipping taken: 14-06-2024 13:25

16. May 23, Quiz 1



Question Label : Multiple Choice Question

3-way-Merge Sort: Suppose that instead of dividing the input list L in half at each step of Merge Sort, you divide L into three equal parts, sort each parts, and finally combine all of them using an efficient three-way merge (merge three sorted lists instead of two).

What is the overall asymptotic running time of the **3-way-Merge Sort** algorithm?

Options :

[redacted] $O(n^2)$

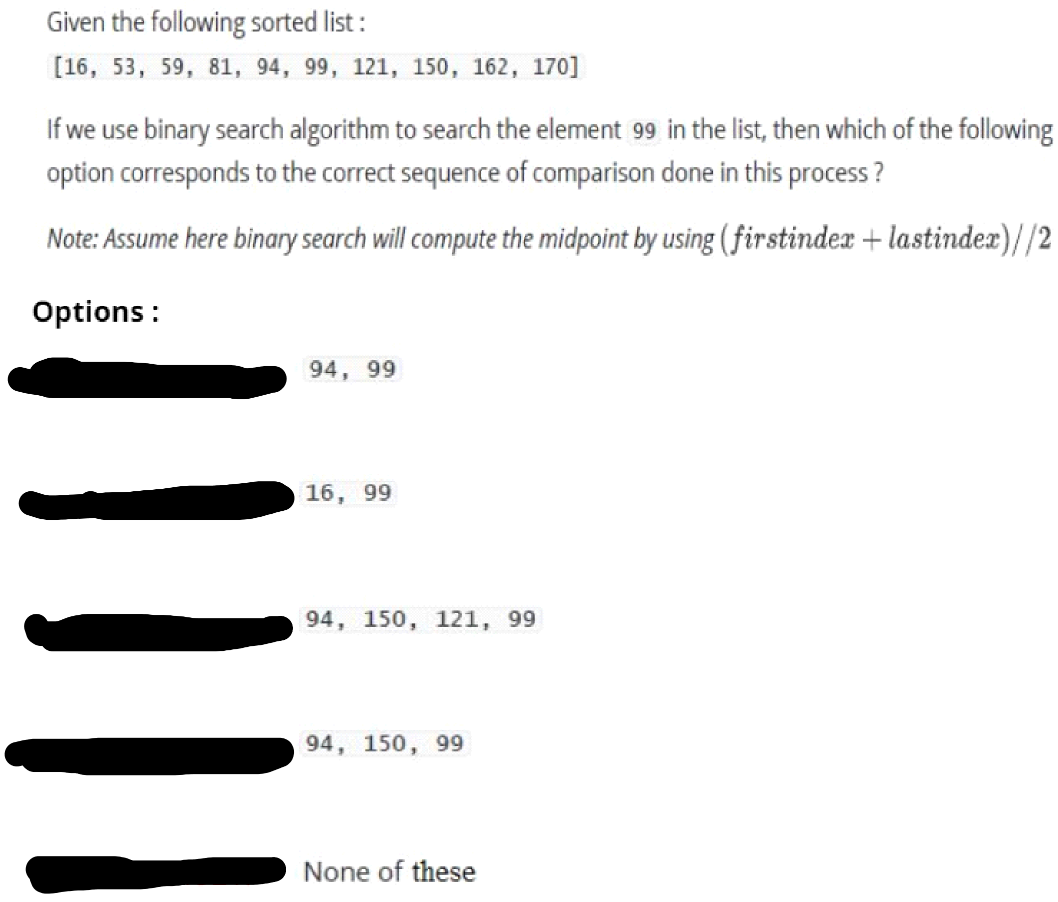
[redacted] $O(n^2 log\, n)$

[redacted] $O(n(log\, n)^2)$

[redacted] $O(n\ log\, n)$

Screen clipping taken: 14-06-2024 13:26

17. Jan 23, Quiz 1



Given the following sorted list :
`[16, 53, 59, 81, 94, 99, 121, 150, 162, 170]`

If we use binary search algorithm to search the element `99` in the list, then which of the following option corresponds to the correct sequence of comparison done in this process ?

Note: Assume here binary search will compute the midpoint by using $(firstindex + lastindex)//2$

Options :

[redacted] `94, 99`

[redacted] `16, 99`

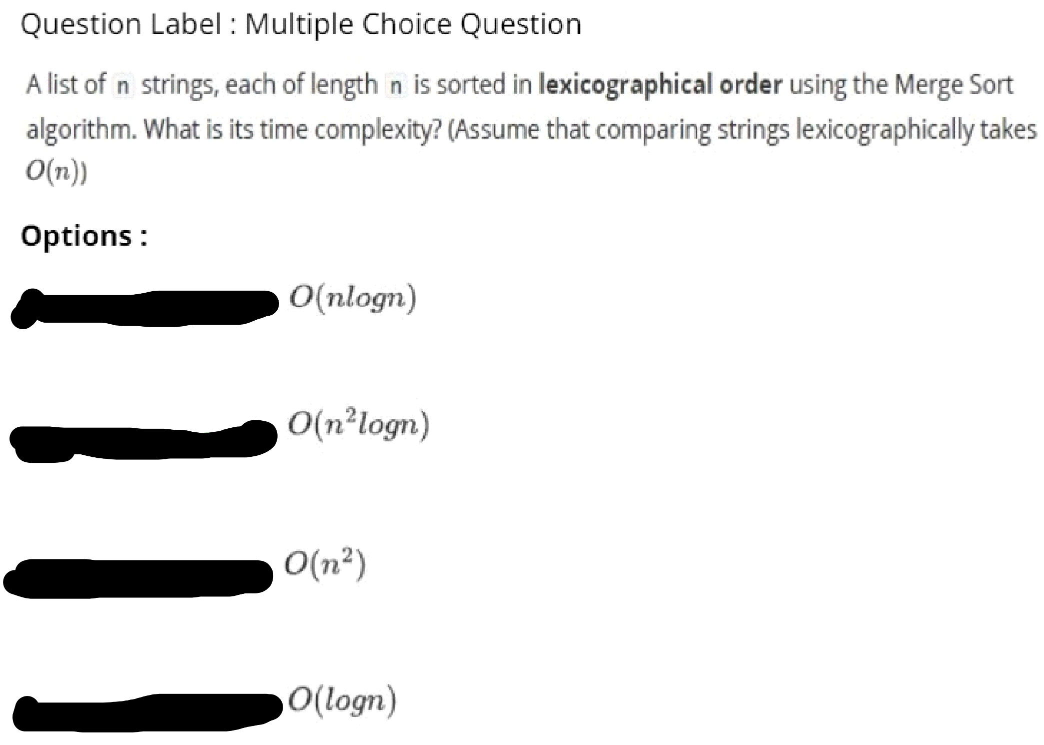
[redacted] `94, 150, 121, 99`

[redacted] `94, 150, 99`

[redacted] None of these

Screen clipping taken: 14-06-2024 13:30

18. Jan 23, Quiz 1



Question Label : Multiple Choice Question

A list of n strings, each of length n is sorted in **lexicographical order** using the Merge Sort algorithm. What is its time complexity? (Assume that comparing strings lexicographically takes $O(n)$)

Options :

[redacted] $O(nlogn)$

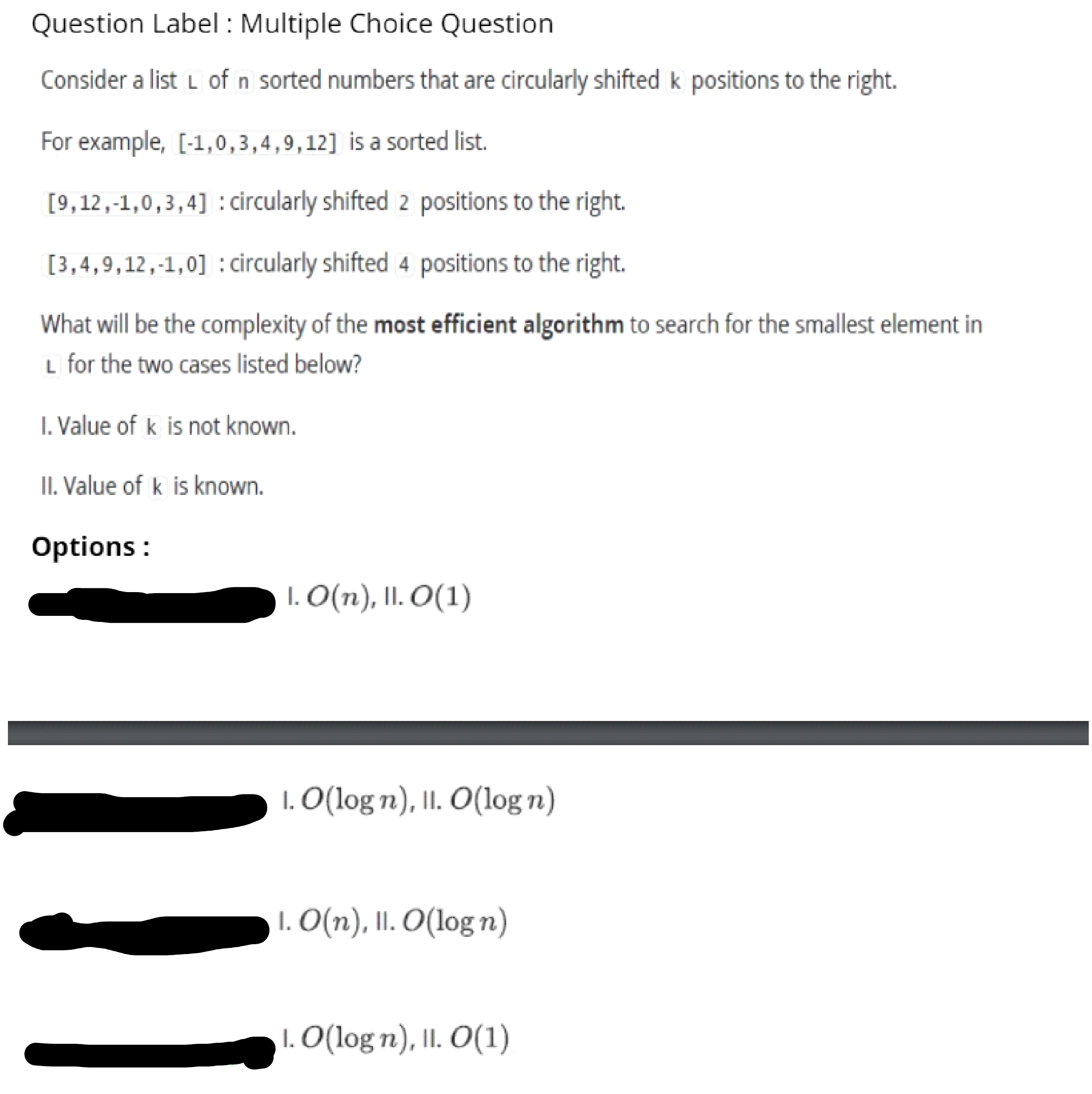
[redacted] $O(n^2logn)$

[redacted] $O(n^2)$

[redacted] $O(logn)$

Screen clipping taken: 14-06-2024 13:31

19. Sep 22, Quiz 1



Question Label : Multiple Choice Question

Consider a list `L` of `n` sorted numbers that are circularly shifted `k` positions to the right.

For example, `[-1,0,3,4,9,12]` is a sorted list.

`[9,12,-1,0,3,4]` : circularly shifted `2` positions to the right.

`[3,4,9,12,-1,0]` : circularly shifted `4` positions to the right.

What will be the complexity of the **most efficient algorithm** to search for the smallest element in `L` for the two cases listed below?

I. Value of `k` is not known.

II. Value of `k` is known.

Options :

[redacted] I. $O(n)$, II. $O(1)$

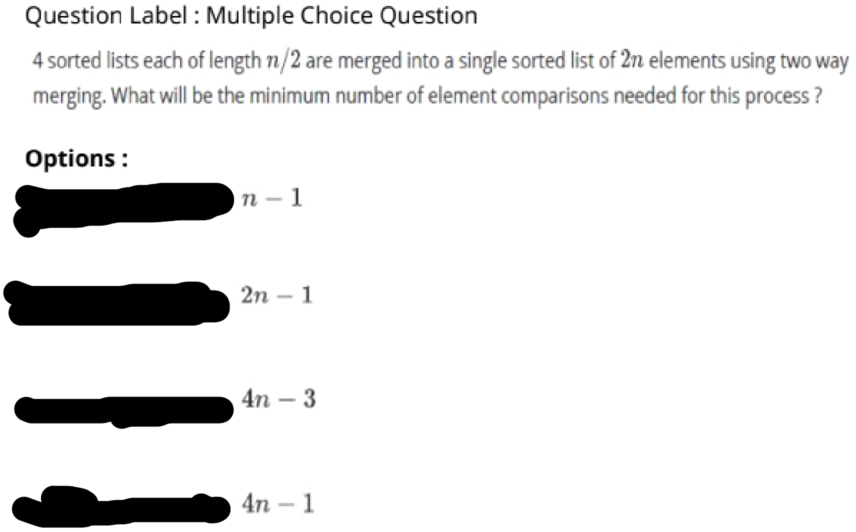
[redacted] I. $O(\log n)$, II. $O(\log n)$

[redacted] I. $O(n)$, II. $O(\log n)$

[redacted] I. $O(\log n)$, II. $O(1)$

Screen clipping taken: 14-06-2024 13:34

20. Jan 23, Quiz 1



Question Label : Multiple Choice Question

4 sorted lists each of length $n/2$ are merged into a single sorted list of $2n$ elements using two way merging. What will be the minimum number of element comparisons needed for this process ?

Options :

[redacted] $n-1$

[redacted] $2n-1$

[redacted] $4n-3$

[redacted] $4n-1$

Screen clipping taken: 06-06-2024 10:06