

Mandatory or Optional :	Mandatory
Number of Questions :	16 16 50
Number of Questions to be attempted :	Yes 0 No
Section Marks :	
Display Number Panel :	
Section Negative Marks :	
Group All Questions :	
Enable Mark as Answered Mark for Review and Clear Response :	Yes
Maximum Instruction Time :	0
Sub-Section Number :	1
Sub-Section Id :	640653107617
Question Shuffling Allowed :	No
Is Section Default? :	null

Question Number : 101 Question Id : 640653737377 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0


Question Label : Multiple Choice Question


THIS IS QUESTION PAPER FOR THE SUBJECT "DIPLOMA LEVEL : PROGRAMMING, DATA STRUCTURES AND ALGORITHMS USING PYTHON (COMPUTER BASED EXAM)"

ARE YOU SURE YOU HAVE TO WRITE EXAM FOR THIS SUBJECT?  
CROSS CHECK YOUR HALL TICKET TO CONFIRM THE SUBJECTS TO BE WRITTEN.

(IF IT IS NOT THE CORRECT SUBJECT, PLS CHECK THE SECTION AT THE TOP FOR THE SUBJECTS REGISTERED BY YOU)

Options :

6406532467847.  YES

6406532467848.  NO

**Sub-Section Number :** 2  
**Sub-Section Id :** 640653107618  
**Question Shuffling Allowed :** Yes  
**Is Section Default? :** null

**Question Number : 102 Question Id : 640653737378 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3 Max. Selectable Options : 0**

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 **false**?

**Options :**

6406532467849.  $f(n) = O(g(n))$

6406532467850.  $g(n) = O(h(n))$

6406532467851.  $f(n) = O(h(n))$

6406532467852.  $h(n) = O(f(n))$

**Question Number : 103 Question Id : 640653737382 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Consider the below **Merge Sort** implementation

```
1 def merge(A,B): # Merge two sorted list A and B
2     (m,n) = (len(A),len(B))
3     (C,i,j) = ([],0,0)
4     #Case 1 :- When both lists A and B have elements for comparing
5     while i < m and j < n:
6         if A[i] <= B[j]:
7             C.append(A[i])
8             i += 1
9         else:
10            C.append(B[j])
11            j += 1
12    #Case 2 :- If list B is over, shift all elements of A to C
13    while i < m:
14        C.append(A[i])
15        i += 1
16    #Case 3 :- If list A is over, shift all elements of B to C
17    while j < n:
18        C.append(B[j])
19        j += 1
20    # Return sorted merged list
21    return C
22
23 def mergesort(L):
24     n = len(L)
25     if n <= 1:
26         return(L)
27     Left_Half = mergesort(L[:n//2])
28     Right_Half = mergesort(L[n//2:])
29     Sorted_Merged_List = merge(Left_Half, Right_Half)
30     return(Sorted_Merged_List)
```

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

**Options :**

6406532467862. Worst case time complexity is  $O(n \log n)$

6406532467863. Best case time complexity is  $O(n)$

6406532467864. Worst case time complexity is  $O(n^2)$

6406532467865. Recurrence relation of merge sort is  $T(n) = 2T(n/2) + O(n)$

6406532467866. It is a stable sort algorithm

**Question Number : 104 Question Id : 640653737383 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

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

- $L[i] > L[i + 1]$  for all  $0 \leq i < m$ , and
- $L[i] < L[i + 1]$  for all  $m \leq i < n-1$ .

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 :**

6406532467867. If  $p > x < q$ , then  $x$  is the minimum in the list.

6406532467868. If  $p < x < q$ , then the minimum element is in the left half of the list.

6406532467869. If  $p < x < q$ , then the minimum element is in the right half of the list.

6406532467870. If  $p > x > q$ , then the minimum element is in the left half of the list.

6406532467871.

If  $p > x > q$ , then the minimum element is in the right half of the list.

**Question Number : 105 Question Id : 640653737385 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

```
1 class Node:
2     def __init__(self,data):
3         self.data = data
4         self.next = None
```

Consider an implementation of a **singly linked list** where each node is created using the given class `Node`. Suppose we have a variable `head` that points to the first node of the linked list.

Which of the below operation **cannot** be performed in **constant time** with the above representation of the linked list?

**Options :**

6406532467876. Insertion of the new node at the front of the linked list.

6406532467877. Insertion of the new node at the end of the linked list.

6406532467878. Deletion of the first node of the linked list.

6406532467879. Deletion of the last node of the linked list.

6406532467880. Deletion of the second node (from starting) of the linked list.

**Question Number : 106 Question Id : 640653737389 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Which of the following is/are possible degree sequence/(s) of vertices of a connected undirected graph with four vertices?

**Note:** Degree sequence is a series of positive integer  $a_1, a_2, \dots, a_n$  where each  $a_i$  is the degree of the  $i^{th}$  vertex of the graph.

**Options :**

6406532467887.     3, 2, 1, 1

6406532467888.     3, 3, 1, 1

6406532467889.     3, 3, 2, 2

6406532467890.     3, 3, 3, 3

**Sub-Section Number :** 3

**Sub-Section Id :** 640653107619

**Question Shuffling Allowed :** Yes

**Is Section Default? :** null

**Question Number : 107 Question Id : 640653737379 Question Type : MCQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3**

**Question Label : Multiple Choice Question**



Consider the following code.

```
1 def fun(n):
2     total = 0
3     for i in range(n):
4         for j in range(n):
5             k = 1
6             while (k < n):
7                 total = total + 1
8                 k = k * 2
9     return total
```

What is the time complexity of the function `fun` in terms of `n`?

**Options :**

6406532467853.  $O(n^2)$

6406532467854.  $O(n \log n)$

6406532467855.  $O(n^2 \log n)$

6406532467856.  $O(n)$

**Question Number : 108 Question Id : 640653737380 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3**

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 :**

6406532467857. `[(9, 0), (6, 1), (8, 1), (5, 2), (7, 5), (2, 5)]`

6406532467858. [(9, 0), (8, 1), (6, 1), (5, 2), (7, 5), (2, 5)]

6406532467859. [(9, 0), (8, 1), (6, 1), (5, 2), (2, 5), (7, 5)]

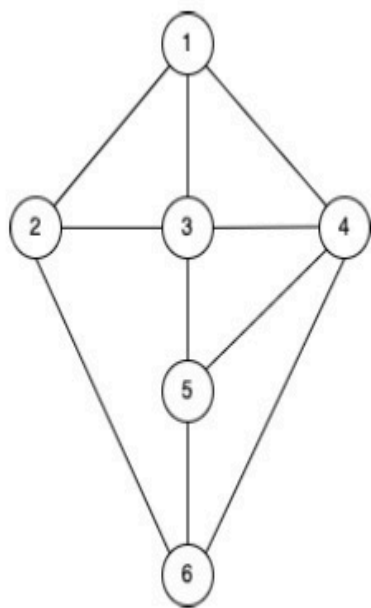
6406532467860. [(9, 0), (6, 1), (8, 1), (5, 2), (2, 5), (7, 5)]

**Question Number : 109 Question Id : 640653737390 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 3**

Question Label : Multiple Choice Question

Consider the following graph



Which of the following vertex sequence is possible **BFS traversals** on the graph started from node 5? Assume that when a node has multiple neighbours, BFS would visit the numerically smaller valued node first.

**Options :**

6406532467891. 5, 4, 3, 6, 1, 2

6406532467892.



5, 3, 4, 6, 1, 2

6406532467893. 5, 3, 4, 6, 2, 1

6406532467894. 5, 6, 4, 3, 2, 1

<b>Sub-Section Number :</b>	4
<b>Sub-Section Id :</b>	640653107620
<b>Question Shuffling Allowed :</b>	Yes
<b>Is Section Default? :</b>	null

**Question Number : 110 Question Id : 640653737384 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4**

Question Label : Multiple Choice Question

Consider the below **Quick sort** algorithm to sort elements in ascending order using first element as pivot.

```
1 def partition(L,lower,upper):
2     # select first element as a pivot
3     pivot = L[lower]
4     i = lower
5     for j in range(lower+1,upper+1):
6         if L[j] <= pivot:
7             i += 1
8             L[i],L[j] = L[j],L[i]
9     L[lower],L[i]= L[i],L[lower]
10    # Return the position of pivot
11    return i
12 def quicksort(L,lower,upper):
13     if(lower < upper):
14         pivot_pos = partition(L,lower,upper);
15         # call the quick sort on leftside part of pivot
16         quicksort(L,lower,pivot_pos-1)
17         # call the quick sort on rightside part of pivot
18         quicksort(L,pivot_pos+1,upper)
19    return L
```

Which of the below input sequence will require the **maximum** number of comparisons?

**Options :**

6406532467872. [22, 25, 56, 67, 89]

6406532467873. [52, 25, 76, 67, 89]

6406532467874. [22, 25, 76, 67, 50]

6406532467875. [52, 25, 89, 67, 76]

**Question Number : 111 Question Id : 640653737386 Question Type : MCQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4**

**Question Label : Multiple Choice Question**

Assume `s` is a stack and `q` is a queue. `Push` and `Pop` operations are usual stack operations, `Enqueue` and `Dequeue` are usual queue operations, and `isEmpty()` is a method that returns true if either the stack or the queue is empty. Assume that stack `s` and Queue `q` are empty initially.

```
1 for i in range(5,0,-1):
2     s.Push(i)
3     q.Enqueue(i)
4
5 while not q.isEmpty():
6     s.Push(q.Dequeue())
7
8 while not s.isEmpty():
9     q.Enqueue(s.Pop())
10
11 while not q.isEmpty():
12     print (q.Dequeue(),end = " ")
```

What is the output of the given code snippet?

**Options :**

6406532467881.    1 2 3 4 5 5 4 3 2 1

6406532467882.    5 4 3 2 1 1 2 3 4 5

6406532467883.    5 4 3 2 1 5 4 3 2 1

6406532467884.    1 2 3 4 5 1 2 3 4 5

**Sub-Section Number :** 5

**Sub-Section Id :** 640653107621

**Question Shuffling Allowed :** Yes

**Is Section Default? :** null

**Question Number : 112 Question Id : 640653737392 Question Type : MCQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 2**

**Question Label : Multiple Choice Question**

Let  $G$  be a graph with  $n$  vertices and  $m$  edges. What is the tightest upper bound complexity of **Depth First search(DFS)** on graph  $G$ , when  $G$  is represented as an adjacency matrix?

**Options :**

6406532467896.  $O(n)$

6406532467897.  $O(n + m)$

6406532467898.  $O(n^2)$

6406532467899.  $O(m^2)$

**Sub-Section Number :**

6

**Sub-Section Id :**

640653107622

**Question Shuffling Allowed :**

Yes

**Is Section Default? :**

null

**Question Number : 113 Question Id : 640653737381 Question Type : SA Calculator : None**

**Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4**

Question Label : Short Answer Question

Consider the following Implementation for **Insertion sort**

```
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)
```

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 :**

**Question Number : 114 Question Id : 640653737387 Question Type : SA Calculator : None**

**Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4**

Question Label : Short Answer Question

**Linear probing** is an open addressing scheme in computer programming for resolving hash collisions in hash tables. Linear probing takes the original hash index and increments the value by 1 until a free slot is found.

A hash table contains 8 buckets indexed from 0 to 7 and uses linear probing to resolve collisions. The key values are integers and the hash function used is  $\text{key} \bmod 8$ . If key values 25, 87, 48, 64, 11 are inserted in to the table in the given order, at what index would the key value 120 be inserted after them?

**Response Type :** Numeric

**Evaluation Required For SA :** Yes

**Show Word Count :** Yes **Answers**

**Type :** Equal **Text Areas :**

PlainText **Possible Answers :**

**Question Number : 115 Question Id : 640653737388 Question Type : SA Calculator : None**

**Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 4**

Question Label : Short Answer Question

An undirected graph  $G$  has 5 vertices. The maximum number of edges in  $G$  is \_\_\_\_.

**Response Type :** Numeric

**Evaluation Required For SA :** Yes

**Show Word Count :** Yes **Answers**

**Type :** Equal **Text Areas :**

PlainText **Possible Answers :**

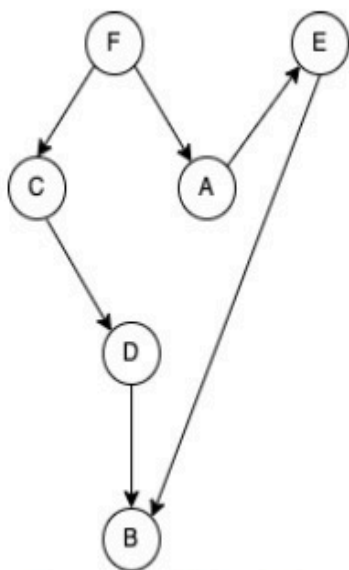
**Question Number :** 116 **Question Id :** 640653737391 **Question Type :** SA **Calculator :** None

**Response Time :** N.A **Think Time :** N.A **Minimum Instruction Time :** 0

**Correct Marks :** 4

**Question Label :** Short Answer Question

Consider the following DAG:



The number of different topological orderings of the vertices of the given graph is \_\_\_\_.

**Response Type :** Numeric

**Evaluation Required For SA :** Yes

**Show Word Count :** Yes

**Answers Type :** Equal

**Text Areas :** PlainText

**Possible Answers :**

## AppDev1

Section Id :	64065351356
Section Number :	8      Online
Section type :	Mandatory
Mandatory or Optional :	16 16 50 Yes
Number of Questions :	0 No
Number of Questions to be attempted :	
Section Marks :	
Display Number Panel :	
Section Negative Marks :	
Group All Questions :	
Enable Mark as Answered Mark for Review and Clear Response :	Yes
Maximum Instruction Time :	0
Sub-Section Number :	1
Sub-Section Id :	640653107623
Question Shuffling Allowed :	No
Is Section Default? :	null

Question Number : 117 Question Id : 640653737393 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0

Question Label : Multiple Choice Question

**THIS IS QUESTION PAPER FOR THE SUBJECT "DIPLOMA LEVEL : MODERN APPLICATION DEVELOPMENT I (COMPUTER BASED EXAM)"**