Best case running time complexity of quick sort _____ best case running time complexity of insertion sort

(a) <

(b) >

(c) =

(d) depends on the input

Worst case running time complexity of quick sort _____ worst case running time complexity of insertion sort

(a) <

(b) >

(c) =

(d) depends on the input

Insertion sort applied to a list of n elements sorted in descending order so as to obtain the output in ascending order would lead to a running time complexity of:

(a) n

(b) n log n

(c) n^2

(d) none of these

Selection sort applied to a list of n elements already sorted in ascending order so as to obtain the output in ascending order would lead to a running time complexity of:

(a) n

(b) n log n

(c) n^2

(d) none of these

A pile of answer scripts (in no specific order) is lying in front of you, and you are required to find the highest marks obtained. The best way to do this would be to:

a) Sort the entire pile and then report the marks obtained on script at the extreme end.

b) Start at the middle of the pile, and decide whether you need to check the upper half of the pile or the lower half; repeat this recursively.

c) Start looking from the top of the pile and keep track of the highest marks obtained thus far; report the highest marks after all the scripts have been examined.

d) Two or more schemes given above are equally efficient.

Consider the following code:

```
for i in range(m):
    for j in range(n):
        // task 1
        // task 2
```

Given that task 1 takes n steps and task 2 takes m steps, the time complexity of the above code is:

(a) $m^2 + n^2$

(b) mn

(c) $(mn)^2$

(d) $mn^2 + m^2n$

The number of bytes of storage required by one pixel of an RGB image is:

(a) 3

(b) 8

(c) 24

(d) 1

Adding 2 to 127 in an 8-bit 2’s complement system would yield:

(a) 127

(b) 0

(c) -127

(d) -128

What would be the output of the following code:

```
def stepup(n):
    while n < 10:
    n += 1
    yield n
for num in stepup(5):
    print(num, end=" ")
```

(a) 6

(b) 5 6 7 8 9

(c) 6 7 8 9 10

(d) 6 7 8 9

What would be the output of the following code?

```
my_list = [11, 22, 33, 44, 55]
my_iter = iter(my_list)
print(next(my_iter))

for i in my_iter:
    if i%2 != 0:
    print(i, end=" ")
```

(a) 11 33 55

(b) 11
33 55

(c) 11
11 33 55

(d) 11 22 33 44 55
11 33 55