DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY,
CSJM UNIVERSITY, KANPUR

Subject- Data Structure Using Python (CSE-S208) Branch CSE-AI

Semester- 2023-2024 (Even Semester) Year: **II Year, Semester IV**

First Mid Semester Examination, 13/02/2024, Shift-II

Time: 1:30 hours Maximum Marks: 30

Note: All questions are compulsory.

Section A

Each question in this section carries 01 marks

1. Exception handling in Python deals with:

   (a) Syntax error

   (b) Semantic error

   (c) Runtime error

   (d) Memory error

2. The fastest stable sorting technique for a large, sorted list is:

   (a) Quick sort

   (b) Merge sort

   (c) Insertion sort

   (d) Selection sort

3. The time complexity to find third largest element from a sorted list (array) of size n with a most efficient algorithm is:

   (a) $O(n^3)$

   (b) $O(n^2)$

   (c) $O(n)$

   (d) $O(1)$

4. Imagine a spreadsheet (table) of student's info of BTech CSE AI is sorted based on roll number of the students as shown below.

RollNo	FirstName	LastName	DSA Score
1	Ankita	Das	30
2	Suraj	Mandal	25
3	Suvam	Basak	27

What would be the best search strategy in term of complexity to find the roll number of the student scored 30 in the course Data Structures?

(a) Binary search

(b) Linear search

(c) Both (a) and (b) will lead to same performance

(d) None

5. Let f and g be functions of natural numbers given by **f(n) = n** and **g(n) = n^2**. Then which is the following statements is/are True:

   (a) f = O(g)

   (b) f = Ω(g)

   (c) f = θ(g)

   (d) None

6. Consider the following function `fun`:

```
def fun(n: int) -> int:
   sum, i= 0, 0
   while i < n:
      i *= 2
      for j in range(n):
         sum += 1
return sum
```

   Which of the following notation is not valid for the given function

   (a) θ(n log n)

   (b) O(n^2)

   (c) Ω(n log n)

   (d) Ω(n^2)

7. Which one is the correct recurrence relation of binary search technique?

   (a) T(n) = T(n-1) + C

(b) T(n) = T(n-2) + C

(c) T(n) = 2T(n/2) + C

(d) T(n) = T(n/2) + C

8. Consider following two line of Python code:

```
email_id = 'suvambasak@csjmu.ac.in'
print (email_id.split('@')[1][:5])
```

What would be the output of this code snippet?

(a) 'suvambasak'

(b) 'csjmu.ac.in'

(c) 'csjmu'

(d) 'ac.in'

9. Consider following function

```
def fun(**args):
   print(type(args))
```

What is the expected output of function call: `fun(name='csjmu')`?

(a) <class 'str'>

(b) <class 'list'>

(c) <class 'dict'>

(d) <class 'set'>

Section B

Each question in this section carries 03 marks

10. The binary search strategy could be implemented with the iterative approach as well as a recursive approach. Which type of implementation would be considered more efficient and why? (1+2)

Iterative implementation will be more efficient in term of memory complexity. Because of no use of stack.

11. Write a function in Python to compute GCD of two number (num_1, num_2) using iterative Euclidean algorithm.

```
def gcd(m: int, n: int):
    if m < n:
        m, n = n, m
    while m % n != 0:
        m, n = n, m%n
```

```
    return n
```

12. How would you utilize the partition function of Quick Sort techniques to find the k-th smallest element in a list? Explain with a few lines of code/algorithm. Also, mention the best and worst-case time complexity of this approach. (2+1)

    Note: Assume that the partition function is given and uses the list's first element as the Pivot element

```
def k_th_smallest(data, l, h, K):
    _k = partition(data, l, h)
    while _k != K:
        if _k > K:
            h = _k-1
        elif _k < K:
            l = _k+1
        _k = partition(data, l, h)
    return _k
```

Best case TC: O(n)
Worst case TC: $O(n^2)$

Section C

Each question in this section carries 06 marks

13. Considers the following Python program below:

```
def fun(n):
   if n <= 1:
      return n
   else:
      x = fun(n-1)
      for i in range(n):
           x += 1
      return x
```

Derive the recurrence relation of the problem. Find the time complexity of the program. (3+3)

Recurrence relation

T(n) = T(n-1) + n + c

T(1) = 1

Time complexity

T(n) = T(n-1) + n + C

= [T(n-2) + n-1 + C] + n + C

= T(n-2) + (n-1) + n + 2C

= [T(n-3) + n-2 + c] + (n-1) + n + 2C

= T(n-3) + (n-2) + (n-1) + n + 3C

…

…

…

= T(n-k) + (n-(k-1)) + (n-(k-2)) + … + (n-3) + (n-2) + + (n-1) + n + kC

= T(1) + 1 + 2 + 3 + … + (n-3) + (n-2) + + (n-1) + n + kC = n(n+1)/2 = $O(n^2)$

14. Derive the worst-case time complexity of Quick Sort from a recurrence relation. What would be your strategy to deal with that, and why? (3+1+2)

Worst-case time complexity of Quick Sort

T(n) = T(n-1) + n + c

T(1) = 1

TC: $O(n^2)$

Strategy: Randomized quick sort

Why: Still the TC is $O(n^2)$ but chance of occurring such case for large n is almost zero because in each pass probability of selecting worst Pivot is (2/n).
