Indian Statistical Institute

BSDS, First Year, First Semester Examination, 2024-25

Introduction to Computing

Full Marks: 50 Date: 18-12-2024 Time: 3 Hours

Answer any *five* of the following questions

5 × 10 = 50

1. Write down the output of the following Python code segments. Justify your answer.

(a)
```
a = 0o10
if a | 7:
     print ("Value =", end = " ")
print (a)
```

(b)
```
n = 20
for i in range(n%2, n//2, -~1):
     print (i, end = " ")
```

(c)
```
def PASS(a = "malay", b = "alam"):
     print (a + b)
PASS("alam")
```

(d)
```
set1 = {1, 3, 5, 7, 9}
set2 = {2, 4, 6, 8}
print (set1 ^ set2)
```

(e)
```
a = [[1, 3, 2], [1, 3, 4, 7], [8, 16, 2, 1, 0, -3]]
sortList = lambda x: (sorted(i) for i in x)
pick = lambda x, f : [y[len(y)-2] for y in f(x)]
result = pick(a, sortList)
print(result)
```

[2+2+2+2+2]

2. Show appropriate examples (pseudocodes as applicable) of the following things in Python:
(i) Use of *constructor*.
(ii) *Operator overloading*.
(iii) *Multiple inheritance*.
(iv) *Method overriding*.
(v) Use of a *Dunder* method.

[2+2+2+2+2]

3. (a) An *n*-digit number is *very special* if the multiplication of its sum of the digits and the product of its digits equals to the original number. For example, 135 is a *very special* 3-digit number. The following erroneous Python function aims to verify whether a number is *very special* or not. Correct it and justify your answer.

```
def VERYSPECIAL(num):
     sum, prod = 0, 0
```

```
    for d in num:
        sum += d
        prod *= d
    if sum * prod == num:
        return("VERY SPECIAL")
    else:
        return("NOT VERY SPECIAL")
```

In what data format the argument is to be passed to the `VERYSPECIAL()` function?

(b) What is the purpose of the following Python code? Justify your answer.

```
def result(string, n):
    for i in range(n):
        print(string[i], end = " ")
def function(s, n):
    for i in range(1, n):
        temp = s[i]
        j = i - 1
        while j >= 0 and len(temp) < len(s[j]):
            s[j + 1] = s[j]
            j -= 1
        s[j + 1] = temp
arr = ["Python", "I", "loving", "am"]
n = len(arr)
function(arr, n)
result(arr, n)
```

[(3+1)+6]

4. The following Python function verifies whether a string, say `s1`, can be obtained by rotating (clockwise or anti-clockwise) another string, say `s2`, by exactly two places or not. For example, "computing" can be obtained from "mputingco" by anti-clockwise rotation by 2 places. Write an efficient version of this code. You will be credited more if your program is more efficient.

```
def isRotated(s1, s2):
    if len(s1) != len(s2):
        return False
    if len(s1) <= 2 or len(s2) <= 2:
        return s1 == s2
    clock = ""
    anticlock = ""
    length = len(s2)
    anticlock = s2[-2:] + s2[:-2]
    clock = s2[2:] + s2[:2]
    return s1 == clock or s1 == anticlock
```

[10]

5. (a) Convert the following infix expression into the respective postfix expression. Consider the priority of operators as '/' > '*' > '+' > '–'.

$$(7 - 3) * (1 + 2) / 3$$

Show the conversion step by step by using an appropriate data structure.

(b) Suppose the in-order and pre-order traversal results obtained from a binary tree are given below.

In-order: 7, 3, 2, 5, 11
Pre-order: 2, 3, 7, 5, 11

Based on the above, construct the original binary tree. Explain each step of construction. [5+5]

6. (a) Show that the running time of *Quicksort* algorithm is $O(n^2)$ when the input array is sorted in descending order and contains distinct elements therein.

(b) Suppose that the partitioning algorithm always produces a 9-to-1 proportional split. What is the runtime complexity of *Quicksort* in this case? Justify your answer. [5+5]

7. Let *Build-Max-Heap* be the algorithm that constructs a Max Heap from an unsorted array. Show that an upper bound on the running time of *Build-Max-Heap* is $O(n \log n)$. Can we derive any tighter bound? If yes, derive that bound. [5+5]
