SRM Institute of Science and Technology, Kattankulathur, Chennai -

603203 Department of Computing Technologies

18CSC207J – Advanced Programming Practice

Cycle Test 1

Answer Key

Part – A

SET - C

1. D. f
2. D.11.0
3. B. ord(S[1])
4. D. [1,2,3,4,5,6]
5. C.8

Part – B

6. Unique Elements Printing:

```
def unique_list(l):
 x = []
 for a in l:
   if a not in x:
     x.append(a)
 return x

print(unique_list([1,2,3,3,3,3,4,5]))
```

7. Yes. The argument count in the function call and function invocation can be differ when default parameter used in the function definition.
   Example:

```
def add(a,b=5,c=10):
   return (a+b+c)
add(3)
```

8. Soln;
   Since python doesn't support constructor overloading, the second constructor definition will hide the previous definition. The instance for the class is always created with default constructor, so the invocation of parameterized constructor results in error.

```
ob2=Student("Test",123) // replace with ob2 = Student()
```

9. Keyword Argument:
   Keyword arguments (or named arguments) are **values that, when passed into a function, are identifiable by specific parameter names**. A keyword argument is preceded by a parameter and the assignment operator, = . Keyword arguments can be likened to dictionaries in that they map a value to a keyword.

10. Constructors are generally used for instantiating an object. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is

created. In Python the __init__() method is called the constructor and is always called when an object is created.
Any simple program using __init__.

Part C

11a. Answer:

(i)

```
list1 = [25,65,21,28]
list2 = [10,16,17,25]
n1=[]
n2=[]
for i in list1:
    if(i%7==0):
        n1.append(str(i))
for j in list2:
    if(j%5==0):
        n2.append(str(j))
list3 = n1+n2
res = [eval(i) for i in list3]
print("Result: ", res)
```

(ii) A dictionary can contain another dictionary, which in turn can contain dictionaries themselves, and so on to arbitrary depth. This is known as nested dictionary.

Example:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'}, 2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
print(people)
```

11. b.

Answer:

```
class Student:
    def __init__(self,name,regno):
        self.name=name
        self.regno=regno
        self.scores = list()
        self.total=0
        self.read()
        self.calculateTotalScore()
    def read(self):
        for i in range(5):
            n = int(input(" "))
            self.scores.append(n)
    def calculateTotalScore(self):
        for i in range(5):
```

```
            self.total =self.scores[i]
def highScore(s,ob):
    count=0
    for o in ob:
        if s>o.total:
            count = count +1
    print(count)

Students = []
ob1=Student('raj',101)
ob2=Student('raju',102)
ob3=Student('ajay',111)
Students.append(ob1)
Students.append(ob2)
Students.append(ob3)
highScore(400,Students)
```