BSCCS2005: Graded Assignment with Solutions
Week 1

1. Match the following:

[MCQ:2points]

A. Control Link	I. Region of the program where a variable is available for use
B. Activation Record	II. Pointer to the previous activation record
C. Scope	III. Duration/time during which a variable is available in the memory
D. Lifetime	IV. Stores the local variables

- ◯ A-II, B-I, C-IV, D-III
- [✓ A-II, B-IV, C-I, D-III]
- ◯ A-II, B-IV, C-III, D-I
- ◯ A-I, B-IV, C-III, D-II

2. What will be the output of the following Python code?

[MCQ:2points]

```
class Demo:
    def __init__(self,str):
        self.name = str
    def print_Demo(self):
        print(self.name)

obj1 = Demo("IITM")
obj2 = Demo("Java")
obj1.print_Demo()
obj2.print_Demo()
```

√
```
IITM
Java
```

◯
```
IITM
IITM
```

◯
```
Java
Java
```

◯
```
Java
IITM
```

3. What will be the output of the following Python code?

[MCQ:2points]

```
i = 42
def f():
    j = i+10
print(i)
f()
print(j)
```

- ◯ 42
  52
- ◯ 42
  0
- [✓ 42 followed by an error]
- ◯ Error

4. Consider the Python code given below and choose the correct option.

[MCQ:2points]

```
def fun1(x):
    y = x+1
    def fun2(z):
        z = ((x+y)*(x-y))*z
        print(z)

    fun2(2)

fun1(5)
```

- ◯ x, y, z are stored in the same activation record.
- ✓ x and y both are stored in the same activation record, whereas z is stored in another activation record.
- ◯ x, y, z are each stored in different activation records.
- ◯ None of the above

Solution: In above code x, y are stored in activation record for function *fun1()* whereas z is stored in activation record of function *fun2()*

5. Consider the statements given below.
Statement 1: **Player** is an object that has name, age and role, as its data.
Statement 2: **Captain** is an object that has name, age, role, date of appointment as a captain, and number of years of experience, as its data.

Identify the correct option regarding subtyping with respect to **Player** objects and **Captain** objects.

[MSQ:2points]

- ✓ Captain can be a subtype of Player.
- ◯ Player can be a subtype of Captain.
- ◯ Captain cannot be a subtype of Player.
- ✓ Player cannot be a subtype of Captain.

Solution: Whatever the data stored by player object, same data also required to the captain.
Here, captain is also a player. Apart player, some extra data is required to the captain.
Captain object can reuse the data from player object by making captain object as subtype of player.
You cannot make player object as subtype of captain because all captain data may not be required to the player object.

6. Consider writing a 'low level' machine language program to perform the following operation:

$$d = a - b * c$$

Identify the correct order to execute the steps given below to perform the given operation.

1. Load the value from memory location a into register `R2`.
2. Load the values from memory locations b and c into registers `R1` and `R2` respectively.
3. Subtract the content of `R1` from `R2` and store the result back into `R1`.
4. Multiply the contents of registers `R1` and `R2` and store the result back into `R1`.
5. Store the content of `R1` into memory location d.

[MCQ: 2points]

○ `2->1->4->3->5`

[✓ `2->4->1->3->5`]

○ `1->2->4->3->5`

○ `2->4->5->1->3`

Solution: The correct steps are as follows:

1. Load the values from memory locations `b` and `c` into registers `R1` and `R2` respectively.
2. Multiply the contents of registers `R1` and `R2` and store the result back into `R1`.
3. Load the values from memory location `a` into register `R2`.
4. Subtract the content of `R1` from `R2` and store the result back into `R1`.
5. Store the content of `R1` into memory location `d`.

7. Identify the most appropriate segregation of the following features between (A) static and (B) dynamic typing in the context of programming.

    1. A name in the program derives its data type from the assigned value.
    2. Every name needs to be declared with its type in advance.
    3. An uninitialized name has no type.
    4. A name cannot be assigned to an incompatible value (a value whose type is not compatible with the type of the name).
    5. A name can be assigned to any value, and the type of the value determines the type of the name.
    6. It does not allow any name to be assigned unless it is already declared explicitly with type.
    7. During any assignment, it cannot identify either if it is an assignment for a new name or if it is a reassignment for an existing name.

[MCQ: 2points]

✓ (A) - 2, 4, 6
(B) - 1, 3, 5, 7

○ (A) - 1, 3, 5
(B) - 2, 4, 6, 7

○ (A) - 1, 3, 4, 5
(B) - 2, 4, 7

○ (A) - 2, 4, 6, 7
(B) - 1, 3, 5

Solution: The features for static type:

- Every name need to declared with their types in advance.
- A name cannot be assigned to an incompatible value (a value whose type is not compatible with the type of the name).
- It does not allow any name to be assigned unless it is already declared explicitly with type.

The features for dynamic type:

- A name in the program derives its data type from the assigned value.
- An uninitialized name has no type,

- A name can be assigned to any value, and the type of value determines the type of the name.
- During any assignment, it cannot identify either it an assignment for a new name or it is a reassignment for an existing name.

8. Consider a polyclinic which has a number of doctors. The doctors have schedules for their visiting days and times. Doctors can update their profiles and change their visiting times. Patients need to register in order to seek appointments with the doctors. Doctors provide prescriptions of various medicines to the patients.

   Given the above scenario, match the abstract types with the associated set of operations.

   [MCQ: 2points]

   ◯
   - Doctor type with the operations – add and modify own profile, add and modify visiting timing, seek appointment
   - Prescription with the operation – write prescription
   - Medicine type with the operation – add medicines

   ◯
   - Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription, seek appointment
   - Patient type
   - Prescription type with the operation – write prescription
   - Medicine type with the operation – add medicines

   ✓
   - Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription
   - Patient type with the operation – seek appointment
   - Prescription with the operation – add medicines
   - Medicine type

   ◯
   - Doctor type with the operations – add and modify own profile, write prescription
   - Patient type with the operations – seek appointment, add and modify visiting timing,
   - Prescription with the operation – add medicines

Solution: The appropriate set abstract types associated with most proper related set of operations are:

- Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription
- Patient type with the operation – seek appointment
- Prescription with the operation – add medicines
- Medicine type

9. What will be the value of **num** after execution of the following code?

[MCQ: 2points]

```
def elementSum(n):
    sum = 0
    while (n != 0):
        sum = sum + n % 10
        n = n // 10
    return sum
num = 22
x = elementSum(num)
```

- ○ 0
- ✓ 22
- ○ 4
- ○ 2
- ○ 10

Solution: **num**, **n** are the names identifying to different memory location, so changing **n** has no effect on **num**

10. What will be the value of **Dict** after execution of the following code?

[MCQ: 2points]

```
def updateDict(d):
    d["rollno"] = 12
Dict = {"name" : "John", "Age" : 21}
updateDict(Dict)
```

- ○ {“rollno” : 12, “Age” : 21}
- ○ {“rollno” : 12}
- ○ {“name” : “John”, “Age” : 21}
- ✓ {“name” : “John”, “Age” : 21, “rollno” : 12}

Solution: **Dict**, **d** are referring to the same memory location, so changing **d** also affects **Dict**