BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Final | **Semester**: Summer 2023
Duration: 105 Minutes | CSE 111: Programming Language II | **Full Marks**: 30
No. of Questions: 3 | **No. of Pages**: 4

Name: (Please write in CAPITAL LETTERS)	ID:	Section:

A

✓ Use the back **part** of the answer script for rough work. **No washroom breaks.**
✓ At the end of the exam, put the question **paper** inside the answer script and **return both**.

Question – 1: CO2 [6 Points]

```
1  class A:
2      temp = 3
3      def __init__(self):
4          temp = 5
5          self.y = 6
6          self.sum = self.temp + 1
7          self.temp += 2
8          A.temp += 4
9      def methodA(self, m,  n):
10         x = self.temp + 4 + n
11         self.sum = x + self.temp
12         self.y = self.y + m + (A.temp)
13         print(x, self.y, self.sum)
14 class B(A):
15     def __init__(self):
16         super().__init__()
17         self.temp = self.temp + A.temp
18         self.sum = 5 + self.temp + A.temp
19     def methodA(self, m,  n):
20         x = A.temp + 3 + n
21         self.sum = self.sum + x + self.temp
22         self.y = A.temp - self.y - n
23         print(x, self.y, self.sum)
24     def methodB(self, b):
25         n = self.temp + b.temp
26         B.temp = self.sum + self.y + n
27         b.methodA(n, B.temp)
28         self.methodA(B.temp, n)
```

Illustrate the output of the following statements:

```
a1 = A()
b1 = B()
b1.methodB(a1)
b1.methodA(-2, 1)
```

Output:

Out1	Out2	Out3
		95
	30	130

Question 2: CO3 [12 Points]

Implement the "**Course**" and "**Student**" classes with the necessary properties to produce the given output for the provided driver code. [**Hint:**
1. All the attributes of the "**Course**" class should be **public**.
2. The attribute "**limit**" in the "**Student**" class should be **private** and set to **None** by default. It will be updated in the **setLimit()** method depending on the CGPA of the student.

- Limit will be 2, if CGPA is less than 2.00
- Limit will be 5, if CGPA is greater than 3.50
- Limit will be 4, otherwise

DO NOT set the limit based on CGPA inside the constructor. The other attributes of the "**Student**" class should be **public**.
3. **A student cannot take any courses if the "limit" variable is not set. If the "limit" variable is set, a student can only take courses up to the limit, NOT exceeding that.**
4. The method **addCourses(*)** in the "**Student**" class can take any number of Course objects as arguments.]

Driver Code	Output

`print(f"Number of courses:{Course.total_courses}")`
`print("1-------------------------")`
`c1 = Course("CSE220", "Data Structures")`
`c2 = Course("CSE230", "Discrete Mathematics")`
`c3 = Course("CSE250", "Circuits and Electronics")`
`c4 = Course("MAT120", "Mathematics II")`
`c5 = Course("PHY112", "Principles of Physics II")`
`c6 = Course("STA201", "Statistics I")`
`c7 = Course("ENG102", "English Composition")`
`print("2-------------------------")`
`print(f"Number of courses:{Course.total_courses}")`
`print("3-------------------------")`
`s1 = Student("Alice",3242544,2.39)`
`s2 = Student("Bob",9878686,3.92)`
`s3 = Student("Carol",2346678,1.67)`
`print("4-------------------------")`
`s1.addCourses(c1,c2)`
`print("5-------------------------")`
`s1.setLimit()`
`print("6-------------------------")`
`print(f"Course limit of {s1.name}`
`is:{s1.getLimit()}")`
`s1.addCourses(c4,c5,c7)`
`print("7-------------------------")`
`s1.addCourses(c1,c2)`
`print("8-------------------------")`
`s1.printDetails()`
`print("9-------------------------")`
`s2.setLimit()`
`s3.setLimit()`
`print("10-------------------------")`
`print(f"Course limit of {s2.name}`
`is:{s2.getLimit()}")`
	`Number of courses: 0`
	`1-------------------------`
	`2-------------------------`
	`Number of courses: 7`
	`3-------------------------`
	`4-------------------------`
	`Sorry. Total course limit is not set.`
	`5-------------------------`
	`6-------------------------`
	`Course limit of Alice is: 4`
	`MAT120 added to course list.`
	`PHY112 added to course list.`
	`ENG102 added to course list.`
	`7-------------------------`
	`CSE220 added to course list.`
	`Cannot add CSE230. Limit exceeded.`
	`8-------------------------`
	`Student ID: 3242544`
	`Student Name: Alice`
	`Courses Taken: ['MAT120', 'PHY112',`
	`'ENG102', 'CSE220']`
	`9-------------------------`
	`10-------------------------`
	`Course limit of Bob is: 5`
	`Course limit of Carol is: 2`
	`11-------------------------`
	`CSE220 added to course list.`
	`CSE230 added to course list.`
	`MAT120 added to course list.`
	`PHY112 added to course list.`
	`ENG102 added to course list.`
	`12-------------------------`
	`CSE230 added to course list.`
	`MAT120 added to course list.`

print(f"Course limit of {s3.name} is:{s3.getLimit()}") print("11-------------------------") s2.addCourses(c1,c2,c4,c5,c7) print("12-------------------------") s3.addCourses(c2,c4,c5,c7) print("13-------------------------") s1.printDetails() print("14-------------------------") s2.printDetails() print("15-------------------------") s3.printDetails()	Cannot add PHY112. Limit exceeded. Cannot add ENG102. Limit exceeded. 13------------------------- Student ID: 3242544 Student Name: Alice Courses Taken: ['MAT120', 'PHY112', 'ENG102', 'CSE220'] 14------------------------- Student ID: 9878686 Student Name: Bob Courses Taken: ['CSE220', 'CSE230', 'MAT120', 'PHY112', 'ENG102'] 15------------------------- Student ID: 2346678 Student Name: Carol Courses Taken: ['CSE230', 'MAT120']

Question 3: CO4 [12 Points]

Design the **SlackWorkspace** class, which is derived from the **Workspace** class, with the necessary properties so that the given output is produced for the provided driver code.
[Hints:
1. The channels are created in the **"createChannels()"** method based on the number of sections, which is the last argument passed to the constructor. If no argument is passed for the number of sections, by default it will be 3. **DO NOT create the channels inside the constructor**.
2. Any number of users can be passed to the **"addMembersToChannel()"** method as an argument.
3. A member's username follows the pattern given below,

name_section

So, you need to divide the username into two parts to find out a user's designated channel.
4. A user cannot be added to a workspace that does not exist in the **channels** dictionary.**]**

Driver Code	Output

class Workspace:
  def __init__(self, workspaceName, type):
  self.workspaceName = workspaceName
  self.type = type
  self.channels = {}
  def checkChannelExistence(self, channelName):
  if channelName in self.channels:
  return True
  return False
  def __str__(self):
  return f"Name: {self.workspaceName}\nType: {self.type}\n"
 	
print(f"Total no. of workspaces: {SlackWorkspace.total_workspaces}")
	Total no. of workspaces: 0
	1============================
	CSE111Workspace is created.
	2============================
	Name: CSE111Workspace
	Type: Education
	Total members: 0
	Total Channels:0
	Channel info: {}
	Total no. of workspaces: 1
	3============================
	Section-1 channel created.
	Section-2 channel created.
	Section-3 channel created.
	Section-4 channel created.
	Section-5 channel created.
	4============================
	Name: CSE111Workspace

```
print("1============================")
CSE111Workspace = SlackWorkspace("CSE111Workspace",
"Education",5)
print("2============================")
print(CSE111Workspace)
print(f"Total no. of workspaces:
{SlackWorkspace.total_workspaces}")
print("3============================")
CSE111Workspace.createChannels()
print("4============================")
print(CSE111Workspace)
print("5============================")
john, harry, adil = "John_2", "Harry_10", "Adil_1"
alice, bob = "Alice_2","Bob_3"
print("6============================")
CSE111Workspace.addMembersToChannel(john, harry)
CSE111Workspace.addMembersToChannel(adil)
print("7============================")
del(CSE111Workspace.channels["Section-3"])
print(CSE111Workspace)
print("8============================")
CSE111Workspace.addMembersToChannel(alice, bob)
print("9============================")
print(CSE111Workspace)
print("10===========================")
CSE320Workspace = SlackWorkspace("CSE320Workspace",
"Education")
CSE320Workspace.createChannels()
print("11===========================")
print(CSE320Workspace)
print("12===========================")
print(f"Total no. of workspaces:
{SlackWorkspace.total_workspaces}")
```

```
Type: Education
Total members: 0
Total Channels: 5
Channel info: {'Section-1': [],
'Section-2': [], 'Section-3': [],
'Section-4': [], 'Section-5': []}
5============================
6============================
John added to Section-2 channel.
Sorry Harry! Section-10 channel not
found in CSE111Workspace.
Adil added to Section-1 channel.
7============================
Name: CSE111Workspace
Type: Education
Total members: 2
Total Channels: 4
Channel info: {'Section-1':
['Adil'], 'Section-2': ['John'],
'Section-4': [], 'Section-5': []}
8============================
Alice added to Section-2 channel.
Sorry Bob! Section-3 channel not
found in CSE111Workspace.
9============================
Name: CSE111Workspace
Type: Education
Total members: 3
Total Channels: 4
Channel info: {'Section-1':
['Adil'], 'Section-2': ['John',
'Alice'], 'Section-4': [],
'Section-5': []}
10===========================
CSE320Workspace is created.
Section-1 channel created.
Section-2 channel created.
Section-3 channel created.
11===========================
Name: CSE320Workspace
Type: Education
Total members: 0
Total Channels: 3
Channel info: {'Section-1': [],
'Section-2': [], 'Section-3': []}
12===========================
Total no. of workspaces: 2
```