Lab 10
OOP – BSDS – Spring 2022

Note: Task 01 is necessary to attempt and has marks 60; whereas task 2 & 3 has only 20 marks each.

Task 01: Write aggregated class to store details of course, teacher and students in the course. First, you have to create class, course, teacher and student course detail in separate files. Next, import these classes and create a class to store details of course, teacher and students in the course. The complete details are as under:

Write following classes with hidden/ private data members each in separate file:

- **Teacher:**
  Data Members: *first name, second name*
  Member Functions: *init*, *str*

- **Course:**
  Data Members: *course code, course title*
  Member Functions: *init*, *str*

- **Student_Subject_Record:**

  Data Members: *roll no, mid, final, sessional*
  Class Level Member: *student count*
  Public Member Functions: *init, str, getter function for mid, final and sessional*
  Private Member Functions: *get probabilistic percentage*

  Function Details:

  ***init*:** *set midterm marks randomly in range 0-35*
  *set final term marks 40 * get probabilistic percentage / 100*
  *set sessional marks 25 * get probabilistic percentage / 100*

  get probabilistic percentage: find percentage of midterm marks, midterm marks are passed as parameter to this function. Find upper and lower difference from percentage that is difference of percentage from 100 and 0. Generate a random number in range 0-10. Check if random number is in range 2 to 8, assign percentage in range plus minus half of upper and lower difference, otherwise, assign percentage at random in range 0-100, see example:

  Assume random value is 4:
  return randint(percentage/2, percentage + (100-percentage/2))
  next assume, random value is less than 2 or greater than 8:
  return randint (0, 100)

- **Course_Class_Teacher:**

  Data Members: teacher, course, students
  Public Member Functions: *init*, *str*, get passed students & sort students in descending order

  Function Details:

  ***init*:** with five data members, *course code, course title, teacher first name, teacher second name and number of seconds*. Create objects of course and teacher class by passing corresponding parameters. Declare a list of students by creating ten student objects, student objects can be created without any parameter randomly

  ***str*:** create a string by concatenating with calling *str* function of course & teacher. Next, run loop over students list and call *str* function for each student object. Concatenate complete output in a single string and return

  ***get passed students*:** Run loop over list of students, check if student has total marks greater than equal 50, call *str* function for each passed student, concatenate output and return

  ***sort student*:** Run nested loop, in inner loop compare total marks of adjacent students, swap if first student has lower total marks than second student

See the following main function and related output:

```
def main():
    cct = Course_Class_Teacher('CC-211','Operating Systems', 'Tahir','Ali', 10)
    print (cct)
    print (cct.get_pass_students())
    cct.sort_students_descending()
    print (cct)ef main():
    cct = Course_Class_Teacher('CC-211','Operating Systems', 'Tahir','Ali', 10)
    print (cct)
```

Output:

Course Code: CC-211 Course Title: Operating Systems

Teacher: Tahir Ali

R_No	Mid	Final	Sess	Total
1	19	22	10	51
2	11	14	6	31
3	22	16	13	51
4	8	22	14	44
5	4	20	15	39
6	29	19	10	58
7	1	20	2	23
8	21	13	18	52
9	29	26	16	71
10	22	15	12	49

R_No	Mid	Final	Sess	Total
1	19	22	10	51
3	22	16	13	51
6	29	19	10	58
8	21	13	18	52
9	29	26	16	71

Course Code: CC-211 Course Title: Operating Systems

Teacher: Tahir Ali

R_No	Mid	Final	Sess	Total
9	29	26	16	71
6	29	19	10	58
8	21	13	18	52
1	19	22	10	51
3	22	16	13	51
10	22	15	12	49
4	8	22	14	44
5	4	20	15	39
2	11	14	6	31
7	1	20	2	23

Task 02: Write recursive function to calculate integer cubic root of a number. Write function with second parameter `i` with default value *1*. If `i * i * i` is equal to number return i. If `i * i * i` is greater than number `return i - 1`, otherwise call recursive function with *i+1*.

```
def main():
    print (int_cube_root(1))
    print (int_cube_root(8))
    print (int_cube_root(64))
    print (int_cube_root(61))
    print (int_cube_root(70))
    print (int_cube_root(120))
    print (int_cube_root(200))
```

```
1
2
4
3
4
4
5
```

Task 03: Write recursive function to print index of all pair of values in the list having same values. See the sample output for your understanding:

```
[3, 5, 1, 2, 1, 3, 4, 5, 5, 1]
(0, 5)
(1, 7)
(1, 8)
(2, 4)
(2, 9)
(4, 9)
(7, 8)
```