OBJECT ORIENTED PROGRAMMING

THURSDAY NOVEMBER 16, 2023: 09:30 PM – 5:30 PM]

ASSIGNMENTS – 10 (RP10) CODE: ASSIGN10

NOTES:

i) Create files with the following file naming conventions: If your roll number ends with **abc**, year of admission is 20**23** and assignment code is **Assign10** then, use the file name as follows: Assign092019abc.cpp (use appropriate extension .cpp suitably).

For example, if the roll number ends with 127; year of admission is 2022 & the assignment code is Assign10, then the file name should be **Assign102022127.java**

ii) Strictly follow the file naming convention. Otherwise, it would attract a penalty up to 20%.

PROBLEM: **[Total Marks: 20]**

You must use JAVA to solve these problems using different object-oriented concepts
Note: Use Random numbers as required and only Public Derivation for inheriting classes as necessary.
You must use collections / iterators as and when necessary.

a) [5 marks] Create a base class `Memories`, which is a Friends Network in which users are connected with one another. Create a random network of n users where $n \in [100, 200]$ using a pointer to an array of objects (Users). Create a base class `UserProfile` for each user u_i having the following member variables:

`UserID`: integer [10000, 99999] – unique ID for each user
(OR you may also use $[1, n]$ as the unique IDs)
`Age`: integer [18, 62] – Age of the user u_i
`Interests`: integer [1, 9] where 1 = news; 2 = tennis; 3 = painting; 4 = music; 5 = singing; 6 = running; 7 = travel; 8 = reading; and 9 = coding;
`Friends`: list of integers [10000, 99999] (or $[1, n]$). The list of all friends u_j of a user u_i where $i \neq j$
`Community`: The list of communities c_r that u_i must belong to. This ranges in [1, 20]
`Date`: The date on which the profile was created.
This ranges in [01/01/2010, 31/12/2020] (create and use a Date class)

Now perform the following tasks:

b) [3 marks] Use function overloading to find friends having the same interests. For example, a user u_i may be interested in **1 (news)**, **7 (travel)**, and **9 (coding)**. A friend u_j of u_i is interested in **1(news)**, 3(painting), 6(running), **7 (travel)**, and **9 (coding)**. User u_j has 2 more interests when compared with u_i. Now you have to identify and print interests that are not matching between u_i and u_j. Repeat the same for all friends of u_i. The above method is expected to take user ID of u_i and / or interests of u_i as arguments.

c) [3 marks] User u_i may have k interests and $k \in [1, 9]$. Write a method to find friends of his friends having at least m matching interests where $m < k$.

d) [3 marks] User u_i may wish to extract different types of friends like close friends, mutual friends, native friends, followers, and acquaintances. You may arrive at a derived class with an additional member variable that could capture the type of friendship u_i may have with u_j.
void findFriends(int *x*) // *x* – denotes the type of friendship & prints all friends with this type
void findFriends(int *x,* int *age*)
void findFriends(int *x,* int *age,* int *interests*)

e) [3 marks] User u_i wants to communicate a specific announcement to all his friends having specific interests, say 9 (coding). Write a virtual method to achieve the same.

f) [3 marks] Write a method to find top 5 communities that consist of members having spent at least *yy* years (assume *yy* = 10 as a test case) since their profile was created. Print the users (with the date) of such communities. Use Map to store user and the communities they are involved in.