CS103 – Computer Programming Spring 2018,

Assignment 3

Question #01

Write the definition of a class *person.* Add at least 5 attributes (private/public) to the *person* class. Add the following behavior to the class:

getName: returns name
setName: sets the name to the passed string
getAge: returns age
setAge: sets the age to the passed integer
isMale: returns true or false
isFemale: returns true or false
getOccupation: returns a string
canCook : returns true or false

Create an object p1 of class *person* and access all the methods of the class and show the result of each method call in proper form.

Default initializes all the attributes of the class.

Default values:
Name: null
Age=0
Gender=m (m for male, f for female)
Occupation= student
Cooking= n (n for no, y for yes)

At the end when you exit your main() and know that your class object is now finish their work then write **Destructor** that shows a message "I am destructor ".

Question #02

Write a class named Employee that has the following member variables:

- **name**. A string that holds the employee's name.
- **idNumber**. An int variable that holds the employee's ID number.
- **department**. A string that holds the name of the department where the employee works.
- **position**. A string that holds the employee's job title.

The class should have the following constructors:

- A **default constructor** that assigns empty strings ("") to the name, department, and position member variables, and 0 to the idNumber member variable.
- A **constructor** that accepts the following values as arguments and assigns them to the appropriate member variables: employee's name, employee's ID number, department, and position.

Write appropriate set and get methods to set and retrieve values in these member variables.

Now use these set methods with three different objects to set the class members with the values given below.

Name	**ID Number**	**Department**	**Position**
Shahid Afridi	47899	Accounting	Vice President
Amir Khan	39119	IT	Programmer
Tom Cruise	81774	Manufacturing	Engineer

Display the data for each employee on the console using get methods.

At the end when you exit your main() and know that your class object is now finish their work then write **Destructor** that shows a message "I am destructor ".

Question #03

Write a recursive function to reverse a string. Write a recursive function to reverse the words in a string, i.e., "cat is running" becomes "running is cat".

Question#04

Find Greatest Common Divisor (GCD) of 2 numbers using recursion.

Question#05

Write a recursive code for bubble sort.

Question#06

Write a recursive code to check whether the given string is palindrome or not.

Question#07

School Management

Create a class named School which has following attributes and member functions to set and get them.

- Teacher: a class with following data members and corresponding setters and getters

  Name (string)

  Rank (enum (P for principal, T for teacher))

Subjects (an array of string)

Class (all classes that has been taught)

- Student: a class with following data members and corresponding setters and getters

  Name

  Roll Number

  Age

  Level (current class i.e. 3rd)

- Location (string)

Now build a school with 5 students (array of Students) and 2 teachers (array of teachers), one is principal (who can also teach) and other is a simple teacher. In your client code, populate all five students and two teachers and print them on start of the program with your school name on the top. Ask users if they want to perform any function like

- Student enrollment -> adding a student
- Teacher recruitment -> adding a teacher
- Search Student
- Search Teacher

Introduce these features to your program with proper display of messages and formatting. After addition of any student or teacher, list should be displayed.

All data should be saved in files so next time your program has previous state. Each time file should be opened with append mode so already written data must not destroyed.

Note: Classes should be built on by keeping in mind Reusability.

Question#08

You are required to design and implement a small student's database using classes. The system is supposed to maintain a record of students enrolled at a college/university. Following information is stored for each student.

- ***student_id*** of type *integer*
- ***student_name*** of type *string*
- ***gender*** of type character

- ***semester*** of type *integer*
- ***department*** of type *string*
- ***gpa*** of type *float.*
- ***extra*** of enumerated data type *extracurricularSkill*

(Note: user defined data type ***string*** of 20 characters will be used above. Moreover, ***extracurricularSkill*** can have the following possible values: *SocietyMember, Sportsman, InternetGeek* and *None*.)

You should use dynamic array implementation. You can use static arrays as well. However, **you will be subject to a 25% of the total marks deduction in case you choose to implement it using static arrays**.

The program should enable its user to perform the following operations.

- store the record of all students, each time the size of the array is increased.

2. **A**

ssign dynamic memory for an initial list of five students. Initially, store data for ten students, using assignment statements.

3. **D**

isplay the information of all the students on campus, in the **tabular form.**

4. **U**

pdate the information of an old student (e.g., change his date of gpa, semester and etc.)

5. **A**

dd the information for a new student. The program should be able to handle all the possibilities (i.e., insert at the start, end or anywhere).

6. **S**

earch and display the information of **a particular student** on campus. (The student information will be searched by giving the **full student name**. In case, no student with that name is found, an error message will be displayed on the screen.)

7. **S**

earch and **Display** the information of all the students **of** a **particular semester** and a **particular department**, in the tabular form.

8. **S**

earch and **Delete** the information of an old student from the record, using the ***student_id***. (If no record entered so far, display an error message). The program should be able to handle all the possibilities (i.e., delete from the start, end or anywhere).

9. **S**earch for **duplicated entries** in the database. Once a duplicated entry is found, **Delete** one of the entries from the database**.** (**Note:** In order to verify this function, you must have some duplicated entries in the database).

10. S uppose the total student community can be divided into three groups

- **Nerds**
- **Vibrants**
- **Dumbs**

The **Nerds** group will be composed of students with GPA more than 3.5 and having ***extracurricularSkill*** *IntenetGeek* or *none**.*** The ***Vibrants*** will be the group of people with GPA more than 3.0 and having any ***extracurricularSkill*** other than *none**.*** The ***Dumbs*** will be the group of people with GPA less than 2.0 and having ***extracurricularSkill*** *none*.

Your job is to read the original list of students and extract the above three groups of students and move them to three new lists, **Nerds**, **Vibrants**, and **Dumbs**, respectively. After this operation the original list of students should be left with only those students which belong to none of these groups. (**Note:** In order to verify this function, you must have data for each group in the original database).

Further Instructions

- The number of students in the record is **dynamic**. For the possible demo / viva voce, **your program should already have at least five student records stored in file**.
- If the current size of the dynamic array is not enough to accommodate the **Add** operation, you have to increase the size of the dynamic array. Whenever the array size has to be increased, it can be done in one of the following ways
    - Increase by a fixed amount, e.g., increment by 10. → 10, 20, 30, 40, …..
    - Double the current size of the array. → 10, 20, 40, 80, …..
- You are required to make a function for each possible operation performed on the list. The operations should be executed with the help of an interactive user interface (e.g., a menu showing all the possible operations).
- Example of the tabular form data display is as follows

ID	Name	Gender	Semester	Department	GPA	Extracurricular Skill

20120001	Ammir	M	8	FCSE	3.0	*Sportsman*
20120001	Bilal	M	7	FME	3.5	*None*

. . .

. . .

20120001 Zain M 6 FEE 2.0 *InternetGeek*

Question #09

Perform +,-,*,/ operations on Rational numbers using class and object. The program should ask for two privte data members numerator and denumerator part of two complex numbers, and display the the result in reduced form.

Question #10

Consider a tollbooth on a bridge. Cars passing by the booth are expected to pay a 50 cent toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth manager keeps track of the number of cars that have gone by, and of the total amount of money collected. Model this tollbooth with a class called*tollBooth*. The two data items are a type *int* to hold the total number of cars, and a type *float* to hold the total amount of money collected. A constructor initializes both of these to 0. A member function called *payingCar*() increments the car total and adds 0.50 to the cash total. Another member function, called *nopayCar*(), increments the car total but adds nothing to the cash total. Finally, a member function called *display*() displays the two totals.

WAP in C++ to simulate and test this class. The program should present the user with three options in every run of the program:

1. to simulate the scenario of a car passing and paying the toll
2. to simulate the scenario of a car passing and not paying the toll
3. to display the results of total cars passed and total toll collected

Once the user selects a particular option from above, the program will execute the functionalities related to that particular option. For example, if they select say option one, then your program needs to call the member function payingCar() and so forth.

After this, the user should be asked whether or not they want to repeat the program. If yes, then the three options above needs to be presented again, else the program will terminate.

Input Case to be tested:
Program Run 1: Car passes and pays the toll

Program Run 2: Car passes and pays the toll
Program Run 3: Car passes and pays the toll
Program Run 4: Car passes and does not pay the toll
Program Run 5: Car passes and does not pay the toll
Program Run 6: Display results

Note: You have to use 3 file structure in questions related to classes.