MACHINE PROBLEM 2

ENGG 30.02

Machine Problem 2 will build on what you've created for Machine Problem 1. It will still have the same major blocks (STUDENT REG, TEACHER INFO, CLASSROOM INFO, and CLASS REGISTRATION blocks). To recap, here are the details:

You are going to build a registration system for a school that has a maximum of thirty (30) students per year level. Since this is a very complex system, you are tasked to create its major blocks one by one.

The first major block (STUDENT REG block) deals with gathering a student's information, saving it into a specific file for a student, and saving the list of students in another file. Here are the details of this major block:

1. The STUDENT REG block does three (3) important tasks. The first task gets the student information and creates a student number (ADD STUDENT). The second task allows a user to display all of the students already included in the list (SHOW ALL STUDENTS) or display a student's information (SHOW STUDENT). The third task allows a user to delete a student from the database (DELETE STUDENT).

2. The information that will need to be saved about the student includes their name (*Last Name*, *Given Name*, *Middle Initial*), address (*House #*, *Street Name*, *Subdivision/Building Name, Barangay Name*, *City/Municipality*), and mobile number. (Take note that it's possible for a student to not have a middle initial or a street name or a subdivision/building name.)

3. When registering a student, the STUDENT REG block will assign a student number after the information mentioned in #1 has been submitted. Student number should have a format of 24-XX. Make sure that you display the student number assigned to a student after getting their information.

4. Create a file named *Last name_Given name.txt* which contains the information in #2 and the student number. The contents of the file should look like this:

```
Reyes, Rose Marie A.
Blue Residences, Bgy. Loyola Heights Quezon City
09101234567
Student # 24-01
```

5. After creating the student file, create a student list file named *Student List.txt*. It should be updated every time a new student is added to the database. Its contents should look like this:

```
24-01 Reyes, Rose Marie A.
24-02 dela Cruz, Juan I.
24-03 Bengua, Glen N.
```

6. A user can choose to display the contents of a student's file (SHOW STUDENT) or display the list of all students (SHOW ALL STUDENTS). It should be displayed similar to how it is stored in the files.

7. A user can choose to delete a student from the database (DELETE STUDENT). When this is done, make sure that the student file is deleted, and the student list file is updated.

The second major block (TEACHER INFO block) gets a teacher's information, saves it into a file, and creates a list of teachers in another file. Here are the details of this block:

1. The TEACHER INFO block does three (3) important tasks. The first task gets the teacher's information and creates a teacher number (ADD TEACHER). The second task allows a user to display all of the teachers already included in the list (SHOW ALL TEACHERS) or display a teacher's information (SHOW TEACHER). The third task allows a user to delete a teacher from the database (DELETE TEACHER).

2. The information that will need to be saved about the teacher includes their name (*Last Name*, *Given Name*, *Middle Initial*), address (*House #*, *Street Name*, *Subdivision/Building Name, Barangay Name*, *City/Municipality*), mobile number, and subjects to teach (*Subject*). (Take note that it's possible for a teacher to not have a middle initial or a street name or a subdivision/building name. Also, a teacher can teach up to 10 possible subjects, which are listed in third major block specifications.)

3. When adding a teacher, the TEACHER INFO block will assign an employee number after the information mentioned in #1 has been submitted. Employee number should have a format of 01-XXX. Make sure that you display the employee number assigned to a teacher after getting their information.

4. Create a file named *Last name_Given name.txt* which contains the information in #2 and the employee number. The contents of the file should look like this:

```
Madamba, Joy Alinda R.
CTC 314 PLDT-CTC Building Bgy. Loyola Heights Quezon City
09109876543
Employee # 01-001
Subjects: Programming, Logic circuits, Microprocessors
```

5. After creating the teacher file, create a teacher list file named *Teacher List.txt*. It should be updated every time a new teacher is added to the database. Its contents should look like this:

```
01-001 Madamba, Joy Alinda R.
01-002 Diokno, Chel I.
01-003 Herrera, Ricky O.
```

6. A user can choose to display the contents of a teacher's file (SHOW TEACHER) or display the list of all teachers (SHOW ALL TEACHERS). It should be displayed similar to how it is stored in the files.

7. A user can choose to delete a teacher from the database (DELETE TEACHER). When this is done, make sure that the teacher file is deleted, and the teacher list is updated.

The third major block (CLASSROOM INFO block) gets a classroom's information, saves it into a file, and creates a list of classrooms in another file. Here are the details of this block:

1. The CLASSROOM INFO block does three (3) important tasks. The first task gets the classroom's information (ADD CLASSROOM). The second task allows a user to display all of the classrooms already included in the list (SHOW ALL CLASSROOMS) or display a classroom's information (SHOW CLASSROOM). The third task allows a user to delete a classroom from the database (DELETE CLASSROOM).

2. The information that will need to be saved about the classroom includes the following: *Room number, Building name (complete), Building name (abbreviation), Classroom type*. There are three classroom types: *Laboratory, Small lecture, Big lecture*. There are also three laboratory classroom types: *Chemical, Computer, Mechanical*. If the room is not a laboratory, N/A should be written in the laboratory classroom type field.

3. Create a file named *Building name(abbreviation)_Room number.txt* which contains the information in #2. The contents of the file should look like this:

```
Room number: 219
Building name (complete): PLDT Convergent Technologies Center
Building name (abbreviation): CTC
Classroom type: Laboratory
Laboratory classroom type: Computer
```

4. After creating the classroom file, create a classroom list file named *Classroom List.txt*. It should be updated every time a new classroom is added to the database. Its contents should look like this:

```
CTC 219
CTC 317
F 308
```

5. A user can choose to display the contents of a classroom file (SHOW CLASSROOM) or display the list of all classrooms (SHOW ALL CLASSROOMS). It should be displayed similar to how it is stored in the files.

6. A user can choose to delete a classroom from the database (DELETE CLASSROOM). When this is done, make sure that the classroom file is deleted, and the classroom list is updated.

The fourth major block (CLASS REGISTRATION block) creates a class for a specific subject, adds a teacher to the class, enlists students in the class list and saves all of this information in separate files per class. Here are the details of this block:

1. The CLASS REGISTRATION block does three (3) important tasks. The first task creates the class for a subject (CREATE CLASS). The second task adds a teacher to the class (ADD TEACHER TO CLASS) and enlists students in the class list (ENLIST STUDENT). The user can then display the information inside the class file (DISPLAY CLASS) or display all the classes created (DISPLAY ALL CLASSES). The third task allows the user to delete a class in the database (DELETE CLASS).

2. The information that will need to be saved about the class includes subject (list of possible subjects are programming, drafting, data analysis, circuits 1, OOP, circuits 2, electronics 1, electronics 2, logic circuits, and microprocessors), days (list of possible days are MTh, TF, M, T, W, Th, F, or S), time (timeslots are 8-930, 930-11,11-12:30, 12:30-2, 2-3:30, 3:30-5, 8-11,11-2, or 2-5), and section name (make sure that only capital letters are used as section name). It is possible to have at most two (2) sections of the same class. Creating a class means asking the user for this information and storing it in a file named *Subject_section name.txt*. Assume that each class is required to have 3 hours per week. The contents of the file should look like this when the class file is first created:

```
Subject: Programming
Days: MTh
Time: 8-9:30
Section name: A
```

3. After creating the class file, the user can ADD CLASSROOM TO CLASS by entering name of the classroom and updating the class file to look like this:

```
Subject: Programming
Days: MTh
Time: 8-9:30
Section name: A
Classroom: CTC 219
```

4. Next, the user can ADD TEACHER TO CLASS by entering the name of the teacher and updating the class file to look like this:

```
Subject: Programming
Days: MTh
Time: 8-9:30
Section name: A
Classroom: CTC 219
Teacher: Joy Alinda R. Madamba
```

5. After adding the teacher, a user can ENLIST STUDENTS in the class file one at a time. Make sure that you add the student # and name on the list. Update the class file for every student added and it should look like this (you don't need to alphabetize the list or sort in terms of student #):

```
Subject: Programming
Days: MTh
Time: 8-9:30
Section name: A
Teacher: Joy Alinda R. Madamba
Students:
1. 24-02 dela Cruz, Juan I.
2. 24-03 Bengua, Glen N.
```

6. Once the class file contains all the information and the enlisted students, the name of the class is added to a class list file named *Class List.txt*. (Reminder to only add the class to the class list once all the information is complete.) A user can then display all of the class (DISPLAY ALL CLASSES) or display one class (DISPLAY CLASS). The format for this file should look like this:

```
Programming A
Programming B
Logic circuits C
Microprocessors A
```

7. A user can choose to delete a class from the database (DELETE CLASS). When this is done, make sure that the class file is deleted, and the class list is updated.

Additional details about the program:

1. It is up to you what the user will input when they want to show or delete a student, teacher, classroom or class. Make sure that you are consistent on what you use across functions.
2. Keep in mind the user-friendliness of how you will ask the required information from your user and how you will display the information they requested.
3. Make sure to integrate all these four blocks into one seamless registration software.

The updates to the specifications from MP1 that are needed for MP2 are the following:

1. You will re-implement your data into classes. You should have at least four (4) classes created in your MP2, and it is up to you what those classes are made up of. The basic requirements for these classes are as follows:
    a. Methods are publicly accessible, and variables are private or protected.
    b. Each class has its own constructor.
    c. At least one parent-child class relationship should be implemented.
    d. The characteristics of encapsulation, inheritance, and polymorphism should be implemented in the parent-child classes created.
2. You are also required to use pointers in this implementation. Make sure to highlight in your code where this is located.
3. You still need to use input validation and header files. Your main.cpp file should only contain the main function.
4. For the STUDENT REG block, you will need to make sure that that a student first exists before it can be deleted from the database and that the user is not allowed to register the same student again.
5. For the TEACHER INFO block, you will need to make sure that that a teacher first exists before it can be deleted from the database and that the user is not allowed to register the same teacher again.
6. For the CLASSROOM INFO block, you will need to make sure that that a classroom first exists before it can be deleted from the database and that the user is not allowed to register the same classroom again.
7. For the CLASS REGISTRATION block, the following are now required:
    a. you will make sure that a class first exists before it can be deleted from the database.
    b. you will check that a teacher added by the user to a class is part of the teacher list.
    c. you will check that the student being enlisted in a class is part of the student list file.

For your submission, you will need to submit three (3) files. These are *main.cpp*, *yourgroupnum_header.cpp*, and your documentation *yourgroupnum_mp2.pdf.* Your documentation should be divided into three (3) parts. Part 1 is the discussion of your algorithm, Part 2 is your flowchart or pseudocode, and Part 3 is your actual code. These three files need to be submitted online before 8am, December 2. You can submit late, but you will get 0 points in the oral part of the machine problem as I will need to see you have submitted your code before I can interview you about your submission.

For the oral part of the machine problem, you will only have fifteen (15) minutes for your presentation. Your presentation will be cut short at exactly 15 minutes, and you will get deductions if you did not finish your presentation on time. You may choose to make a live presentation of your submission or play a video presentation where all group members talk. Make sure to include a very short demonstration of your software. It is up

to you to choose the format of your presentation, but the basic goal of your presentation is to show what each member has accomplished and how.

BREAKDOWN OF MP GRADES:
Program – 40 points
Documentation – 20 points (paper) + 20 points (oral)
Efficiency – 20 points

BONUS POINTS (you can only get a maximum of 30 bonus points):
If you would like to get bonus points, you can add the following functionality:

1. you will alphabetize the class list. (10 points)
2. you will sort in terms of student #. (10 points)
3. you will check that a class assigned to a teacher has that subject listed in his/her information and inform the user if not. (10 points)
4. you will check that a student is not enlisted twice in the same subject with different sections. (20 points)
5. you will check that a student did not enlist in different subjects with the same day and time slot (or conflict in schedule). (20 points)