

**EEE 2110**

# **Programming Language Lab**

## **Details of Project and Project submission Rules**

*This Project assesses Course Outcome 04 with Bloom's Taxonomy Psychomotor level 2*

**Marks: 20**

### **Project Submission Rules:**

Submission would be individual through “Google Form” within **02.10.2021**

1. Submission would be in .pdf format with following topics highlighted.
  - a. Introduction to the problem
  - b. The flow chart /way to solve the problem
  - c. The final code
  - d. Screenshot of the solution of the problem
  - e. Conclusion/Discussion
2. Your submission must include a top page including
  1. Topic of your presentation
  2. Name of the presenters
  3. Student ID of the presenter
  4. Lab Group
3. And it should be renamed as { **Roll Number topic name.pdf** } e.g. {190105XXX\_Hospital management system.pdf.}
4. .cpp file need to be submitted in separate section of the google form and it should be renamed as { **Roll Number topic name.cpp** } e.g. {190105XXX\_Hospital management system.cpp.}
5. No late submission is acceptable
6. Any **copy/plagiarized** submission is considered as **0 marks**
7. In case of viva/presentation, computer has to be functional. Any malfunction of computer or laptop or webcam will not be acceptable.

## **Overview of Projects**

| <b>Sl No</b> | <b>Name of the Project</b>                   |
|--------------|--|
| <b>1.</b>    | University Management System                 |
| <b>2.</b>    | Class Routine Management System              |
| <b>3.</b>    | Lab Routine Management System                |
| <b>4.</b>    | Test Cricket                                 |
| <b>5.</b>    | Checker Board Game                           |
| <b>6.</b>    | Sundarban                                    |
| <b>7.</b>    | ATM CARD                                     |
| <b>8.</b>    | Doctors Appointment                          |
| <b>9.</b>    | Mortal Combat                                |
| <b>10.</b>   | Snake Ladder Game                            |
| <b>11</b>    | Hotel Management System                      |
| <b>12</b>    | Library Management                           |
| <b>13.</b>   | Book Shop Management System                  |
| <b>14</b>    | Grading Management System for EEE Department |
| <b>15.</b>   | Stock Market                                 |
| <b>16</b>    | Data System                                  |
| <b>17</b>    | TRAIN  |
| <b>18.</b>   | Canteen Management System                    |
| <b>19.</b>   | Travel Agency Management System              |
| <b>20.</b>   | Pharmacy Management System                   |
| <b>21.</b>   | Pay Roll System                              |
| <b>22</b>    | Hospital Management System                   |
| <b>23</b>    | Banking Account System                       |
| <b>24</b>    | Contact Application                          |
| <b>25</b>    | Student Fees system                          |

# **Project1**

## University Management System

Write a C++ program for maintaining the student's class performance record with the following features.

- a) There will be login for three types of user account – **admin**, **teacher** and **student**.
- b) Admin can register or delete new teacher as well as student account, and create username and password for that account. Default Admin username is : "admin" , password is "@dmin@U\$T"
- c) Students can login with a valid username-password. Each student can mark his/her attendance of the same day. If present the attendance would be '1' otherwise '0'.
- d) Teachers can login with a valid username-password. Then a teacher can mark the performance of the present students of that same day only. The performance mark would be within 0-10.
- e) Students can see the history of previous attendance as well as performance.
- f) Both Admin and teachers can get the summary of student's class performance – individually and collectively.

### **Task Achievement:**

To implement the program the user should follow the following steps:

- Implement a class named **admin** which will have the functions: **createAccount()**, **seePerformance()**
- Implement a class named **teacher** which is comprised of the functions: **givePerformance()**, **seeAttendance()**, **seePerformance()**
- Implement a class named **student** which may have the method like **giveAttendance()**
- Save the database of student with their name, attendance and performance in a text file named as **studentDb.txt**. The file should be look like below:

| Name  | Attendance | Performance |
|-------|------------|-------------|
| Aftab | 1          | 10          |
| Navid | 0          | 0           |
| Atif  | 1          | 9           |

- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this game.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.

**Note:**

- Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
- Students are not allowed to use the concepts beyond the syllabus.
- The programming language is C++. Try to avoid the coding pattern of C.
- While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.

## **Project2**

### **Class Routine Management System**

Write a C++ program for managing the classrooms of the EEE Department with the following features.

- a) There are several classrooms. Each classroom must have their unique number (e.g., 4A04, 5A04, 8A04 etc.). These classroom numbers are saved in a text file named as “**classroom.txt**”. Admin will read the text file and retrieve the room names. Admin can create as many classroom as he wishes but the created classroom cannot exceed the list in the text file.
- b) There will be login for three types of user account – admin, teacher and student.
- c) Admin can register or delete new teacher as well as student account, and create username and password for that account. Default Admin username is: "admin", password is "@dmin@U\$T".
- d) Admin can also create/delete classroom.
- e) Admin can assign courses to the classroom. Program must check that no two (or more) courses are overlapped during the class time. If overlap happens, a warning/error will be displayed to the admin during assigning the classrooms.
- f) Teachers can check the classroom number of his/her classes according to the day of the week.
- g) Students can check the room number, day and time of any specific course by inserting the course number.
- h) The routine of any classroom for all the working days [Sunday- Thursday] can be seen by admin or teacher.

#### **Task Achievement:**

To implement the program the user should follow the following steps:

- Implement a class named **admin** which will have the functions: **createAccount()**, **createClassRoom()**, **scheduleCourse()**
- Implement a class named **teacher** which is comprised of the function: **checkRoutine()**
- Implement a class named **student** which may have the method like **checkRoutine ()**
- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this game.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.

**Note:**

- Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
- Students are not allowed to use the concepts beyond the syllabus.
- The programming language is C++. Try to avoid the coding pattern of C.
- While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.

## Project3

### Lab Routine Management System

Write a C++ program for managing the laboratories of the EEE Department with the following features.

- a) There will be login for three types of user account – admin, coordinator and student.
- b) Admin can register or delete new coordinators as well as student account, and create username and password for that account. Admin can also create Default Admin username is : "admin" , password is "@dmin@U\$T".
- i) Admin can create/delete new laboratories with unique names and numbers (e.g., DSP lab [4B07], Circuit lab [3B02], etc.). These lab-room names and numbers are saved in a text file named as “**labroom.txt**”. Admin will read the text file and retrieve the room names. Admin can create as many lab-room as he wishes but the created lab-room cannot exceed the list in the text file.
- c) Admin can also assign one coordinator for each lab.
- d) For each laboratory, the coordinator can make a list of available devices by adding devices, deleting devices, updating devices, etc. Coordinator can also mark the devices as functional and defective.
- e) Coordinator can assign the routine for each lab class for the semester.
- f) By logging in their accounts, students can check the list of the functional and defective devices as well as class routine for any laboratories by inserting the laboratory name.
- g) The “**labroom.txt**” looks like below:

|             |   |
|-------------|---|
| [CL1- 3B01] | CIRCUIT LAB I                                   |
| [CL2- 3B05] | CIRCUIT LAB II                                  |
| [EL- 3B06]  | ELECTRONICS LAB                                 |
| [PEL- 3B07] | POWER ELECTRONICS LAB                           |
| [ML- 4B06]  | MACHINE LAB                                     |
| [DML- 4B08] | DIGITAL AND MICROPROCESSOR LAB                  |
| [TML- 4B04] | TELECOMMUNICATION AND MICROWAVE ENGINEERING LAB |
| [CSL- 9A06] | CONTROL SYSTEM LAB                              |
| [DSP -4B01] | DSP LAB   |



|               |                    |
|---------------|--------------------|
| [SL- 4B02]    | SIMULATION LAB     |
| [PSL- 4B07]   | POWER SYSTEM LAB   |
| [VLSI-4B03]   | VLSI LAB           |
| [EL II- 4B05] | ELECTRONICS LAB II |

### **Task Achievement:**

To implement the program the user should follow the following steps:

- Implement a class named **Admin** which will have the methods : **createAccount()**, **createLab()**
- Implement a class named **Cordinator** which can function through the methods such as, **addDevice()**, **deleteDevice()**, **updateDevice()**, **makeRoutine()** etc.
- Implement a class named **student** which can see his/her lab routine using **getRoutine()**.
- Implement a class named **device** which will keep and update the device information.
- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this game.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.

### **Note:**

- Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
- Students are not allowed to use the concepts beyond the syllabus.
- The programming language is C++. Try to avoid the coding pattern of C.
- While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.

## Project 04

### Test Cricket

We are familiar with test cricket which consists of two innings. Based on toss TeamA gets fielding where TeamB has to Bat first. Each team has eleven players. Each player has profile (total run, total centuries, total innings played, total wickets, striking rate, bowling rate). Since, it is a computer based game; the decision of each ball played will be determined by random number.

As a case study, a bowler bowls and the batsman play that delivery. Outcome will be the run (0-6) or wicket fall (bold, catch, run, stamped out). Score of the match and the profile of the players will be updated after each delivery. Each innings there are 10 overs and 10 wickets (though 11 eleven players).

|                 |              |
|-----------------|--------------|
| Name            | Amir Ebrahim |
| Age             | 34           |
| BATSMAN         |              |
| Innings         | 153          |
| Runs            | 8752         |
| Highest Score   | 124          |
| Average         | 59           |
| Strike rate     | 70%          |
| 100s            | 16           |
| BOWLER          |              |
| Innings         | 100          |
| Bowling Average | 39.7         |
| Wickets         | 159          |

| MATCH SUMMARY                 |       |         |      |
|-------------------------------|-------|---------|------|
| 1 <sup>st</sup> innings       |       |         |      |
| TeamA                         | 305/8 |         |      |
| HighestScorer1                | 103   | Bowler1 | 4-60 |
| HighestScorer2                | 97    | Bowler2 | 3-50 |
| HighestScorer3                | 81    | Bowler3 | 1-39 |
|                               |       |         |      |
| TeamB                         | 395   |         |      |
| HighestScorer1                | 133   | Bowler1 | 4-60 |
| HighestScorer2                | 91    | Bowler2 | 3-50 |
| HighestScorer3                | 85    | Bowler3 | 1-39 |
| TeamB is 90 runs ahead        |       |         |      |
| 2 <sup>nd</sup> innings       |       |         |      |
| TeamA                         | 405   |         |      |
| HighestScorer1                | 152   | Bowler1 | 4-68 |
| HighestScorer2                | 103   | Bowler2 | 3-55 |
| HighestScorer3                | 69    | Bowler3 | 2-49 |
|                               |       |         |      |
| TeamB                         | 290   |         |      |
| HighestScorer1                | 119   | Bowler1 | 5-46 |
| HighestScorer2                | 61    | Bowler2 | 3-50 |
| HighestScorer3                | 55    | Bowler3 | 2-39 |
| TeamA is 25 runs ahead        |       |         |      |
| TeamA won the test by 25 runs |       |         |      |

### **Task Achievement:**

To implement the program the user should follow the following steps:

1. Implement a class named **ScoreBoard** which will show the latest score of the match.
2. Implement a class named **Player** which will keep and update the scores of the player. Initialization of the player can be done by reading the text file named **“players.txt”** which keeps the record of a player
3. Implement a class named **Cricket** which will play the game with necessary options, scoreboard, players, display menus and result.
4. Implement a class named **SummaryBoard** which will show the Match summary. Besides, it will update the profile of the relevant players who has played the current match.
5. Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this game.
6. Sample **“players.txt”** looks like below:

| Name   | Age | Bat_Innings | TotalRun | HishestScore | Average | StrikeRate | BowInnings | AvgRun/Over | Wickets |
|--------|-----|-------------|----------|--------------|---------|------------|------------|-------------|---------|
| Hamid  | 29  | 100         | 5000     | 100          | 50      | 70         | 90         | 5.1         | 86      |
| Zarir  | 27  | 80          | 3000     | 167          | 38      | 65         | 70         | 4.9         | 57      |
| Rahbar | 32  | 162         | 9654     | 105          | 60      | 63         | 152        | 4.7         | 156     |
| Saleh  | 34  | 169         | 9154     | 109          | 54      | 79         | 139        | 5.2         | 123     |
| Nadeem | 21  | 70          | 2598     | 145          | 37      | 55         | 58         | 4.2         | 43      |

### **Note:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 05

### Checker Board Game

This game is a simple and famous board game. The board looks similar to chess board which has 8x8 grids. For easier visibility grids are colored black and white by turns. Two players take two sides where each side has 12 pieces of disk. One player has dark colored disks and the other has disks of lighter color for easier identification.

#### Rules of the game:

The rules are as follows:

1. Starting layout of the disks of player one ('o') and player two ('\*') is shown in Figure.
2. Disk can move diagonally by one step. If it finds opponent disk it can jump over it diagonally and rest in next diagonal position if the position is blank.
3. A player can march the disk to the end line of the other side according to rule 1. Then, it achieves double power and can move both back and forth direction following rule 1.
4. The loser will lose all his disks.

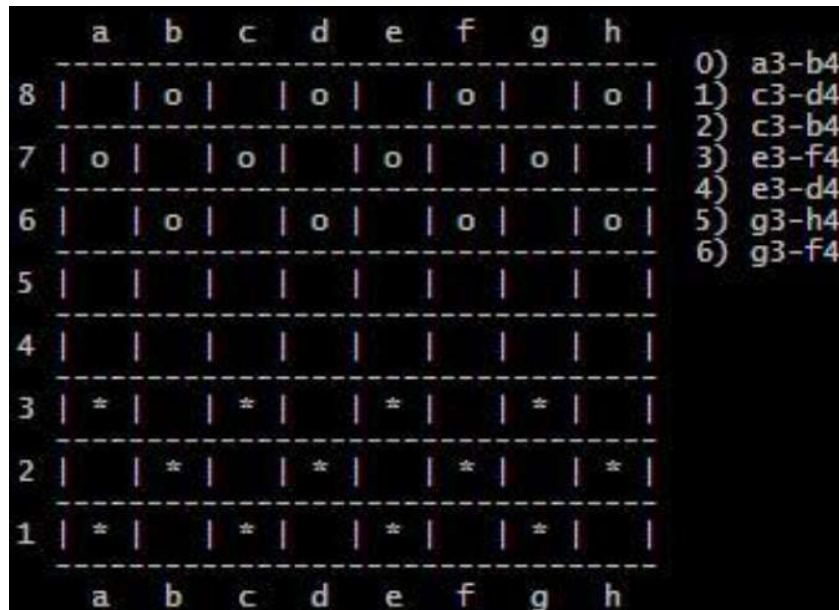





Figure: The layout of Checkerboard Game.

### **Task Achievement:**

To implement the program the user should follow the following steps:

1. Implement a class named **board** which will draw the board and show the disks. You must have several option of board based on the choice of user. The options are
  -  Easy Mode [8x8]
  -  Normal Mode[12x12]
  -  Hard [16x16]
2. Implement a class named **player** which will keep the profile of the player and record of the disks with position.
3. Implement a class named **checker** which will play the game with necessary options, board, players, display menus and result.
4. Implement a class named **ScoreBoard** which will show the name of the player, matches played by him and the number of winning matches. At the end of the program the scoreboard should be saved in a text file named “**ScoreBoard.txt**”.
5. Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this game.
6. The sample text file“**ScoreBoard.txt**” is shown below:

| PlayerName | Number_Playing | Number_Winning |
|------------|----------------|----------------|
| Raihan     | 10             | 7              |
| Rakeeb     | 20             | 9              |
| Razeen     | 25             | 15             |
| Rishat     | 15             | 10             |
| Riaz       | 13             | 6              |

### **Note:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project6

### Sundarban

Write and apply a C++ program that will simulate the Sundarban forest. Consider the forest as a grid of 100 x 100 positions. The four direction of movement is Up, Down, Left, Right in the grid. There are five animals and their moving behavior is as follows:

- Tiger → jumps 5 steps to any direction and move 2 steps to clockwise direction.
- Deer → jumps 6 steps to any direction and move 3 steps to anticlockwise direction.
- Monkey → jumps 7 steps to any direction and move 1 step to clockwise direction.
- Cat → jumps 2 steps to any direction and move 1 step to anticlockwise direction.
- Snake → slither 2 steps to any direction and move 1 steps to clockwise direction again slither 2 steps to previous direction and move 1 steps to anticlockwise direction

The traverse of animal in the forest is based on the following rule:

- At the initial stage of the simulation, generate animals of each category [1-100] randomly with the random position and random direction.
- Then they will move continuously at each round.
- Both of the animals will die if they fall in same grid point.
- After each round the number of alive animals are recorded and shown in the screen with round number.
- The round will go on till all animal die or at least only one remains.
- Since, the generation of animal numbers is random, total number of animals can be odd. In that case, one will remain at the end. If the number is even then all will be dead after several simulations.
- If any animal tries to cross the border of grid point then, its motion will be reversed so that it will again traverse in the forest.
- Recorded living animals with the round number should be saved in a text file named as **“livingAnimals.txt”**.

- The data organization of sample text file is similar as below:

| Round | Tiger | Deer | Monkey | Cat | Snake |
|-------|-------|------|--------|-----|-------|
| 1     | 25    | 75   | 59     | 54  | 35    |
| 2     | 20    | 70   | 50     | 50  | 30    |
| 3     | 17    | 67   | 43     | 46  | 27    |
| 4     | 13    | 63   | 33     | 41  | 22    |

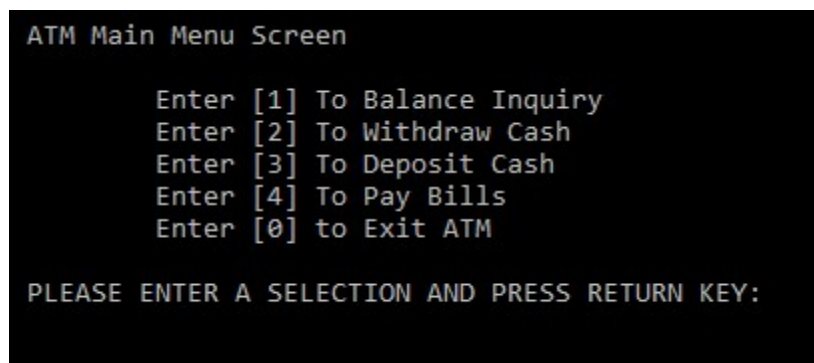
**Note:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 07

### ATM CARD

An ATM card is a payment card or dedicated payment card issued by a bank which enables a customer to access their financial accounts via its Automated Teller Machine (ATM). Each ATM card must have a PIN number to access it.



Layout of ATM

#### Task Achievement:

To implement the program the user should follow the following steps:

1. Implement a class named **Bank** which will keep all information of account holder (Account holder's name, Address, Branch, Account number, Current balance). Initiate the instance of Bank with the text file "**Bank.txt**". This file has keeps the information of some account holders.
2. Implement a class named **ATMcard** which will have a PIN number and will keep all information of account holder.
3. Implement a class named **ATMbooth** which will have necessary options (Balance Inquiry, Withdraw Cash, Deposit Cash, Pay Bills, Exit ATM)
4. Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this ATM.



5. The text files have the data organization as below: Account holder's name, Address, Branch, Account number, Current balance

| AccountNumber | Name   | Address | BranchCode | CurrentBalance |
|---------------|--------|---------|------------|----------------|
| 101200        | Naim   | Dhaka   | 101        | 50000          |
| 105100        | Nazim  | Khulna  | 105        | 10000          |
| 110500        | Naveed | Sylhet  | 110        | 53000          |
| 112322        | Nabeel | Natore  | 112        | 83000          |
| 115425        | Nafee  | Jessore | 115        | 15000          |

But in the text file you can avoid the first row to make it simpler. Just Keep the data and avoid header. You can keep the data in a structure since different data field has different data type. The data in the text file looks like below:

|        |        |         |     |       |
|--------|--------|---------|-----|-------|
| 101200 | Naim   | Dhaka   | 101 | 50000 |
| 105100 | Nazim  | Khulna  | 105 | 10000 |
| 110500 | Naveed | Sylhet  | 110 | 53000 |
| 112322 | Nabeel | Natore  | 112 | 83000 |
| 115425 | Nafee  | Jessore | 115 | 15000 |

**Note:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 08

### Doctors Appointment

#### Rules:

1. If Dr. A is booked at any chosen time then search Dr. B to know either available or booked at that time. If Dr. B is available at that time then take decision either booking Dr. B or see another schedule to book Dr. A. Same things can be done for all schedule and all days for both Dr. A and Dr. B.
2. If a patient books a doctor then do registration process and save in a file.

```
A Medical College and Hospital
-----

Select one of the following doctors:
-----

1: Dr. A
2: Dr. B

Select one of the following days:
-----

0: Monday
1: Tuesday
2: Wednesday
3: Thursday
4: Friday

Select one of the following times:
-----

0: 10 AM - 11 AM
1: 12 AM - 1 PM
2: 2 PM - 3 PM
3: 4 PM - 5 PM

Registration:
-----
Name      :
Age       :
Cell No.  :
```

#### Task Achievement:

To implement the program the user should follow the following steps:

1. Implement a class named **Hospital** which will keep all information of doctors (Name, Day, Time slots, Available or Booked).
2. Implement a class named **Doctor** which will keep all information of Dr. A (Name, Day, Time slots, Available or Booked). Initialize the object with “drA.txt” which keeps the schedule of Dr.A

3. Similar object should be created for Dr.B with the initial data coming from “**drB.txt**”.
4. Implement a class named **Registration** which will have necessary input options to register.
5. Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this appointment.
6. The text files have the data organization as below:

| DAY | SLOT▶ | 0 | 1 | 2 | 3 |
|-----|-------|---|---|---|---|
| ▼   |       |   |   |   |   |
| 0   |       | Y | N | N | N |
| 1   |       | N | Y | N | Y |
| 2   |       | N | N | Y | N |
| 3   |       | N | N | N | N |
| 4   |       | Y | Y | Y | N |

But in the text file you can avoid the first row and first column to make it simpler. Just Keep Y and N. By the position of the Y and N, you can infer the data and keep it in an array to use later.

The data in the text file looks like below:

|   |   |   |   |
|---|---|---|---|
| Y | N | N | N |
| N | Y | N | Y |
| N | N | Y | N |
| N | N | N | N |
| Y | Y | Y | N |

Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 09

### Mortal Combat

Make a simplified version of Mortal Combat game, where the character list with their features are stored in the file named **MK.txt**. The data in the file looks like below:

| <b>CharacterName</b> | <b>BlockPower</b> | <b>AttackPower</b> | <b>SpecialPower</b> |
|----------------------|-------------------|--------------------|---------------------|
| Subzero              | 10                | 12                 | 30                  |
| Scorpion             | 9                 | 16                 | 25                  |
| Raiden               | 15                | 20                 | 40                  |
| Jaks                 | 11                | 13                 | 23                  |
| Kitana               | 13                | 13                 | 29                  |

Any player starts the game with the life of 100 points. When he/she **[p1]** attacks someone **[p2]** the damage to other player **[p2]** will be with the **AttackPower** of p1. But the opponent **[p2]** can block this attack. If the opponent **[p2]** can block, then the damage will be = **AttackPower [p1] – BlockPower [p2]** . Instead of Attack, superAttack can be performed randomly.

- At first the user will give his/her name and the choice of the **CharacterName**.
- Then the game will start with a computer player.
- The computer player will take the first **CharacterName** from the list. This is round1. If the human player win, then the game will continue to round 2 and the computer player will take second **CharacterName**.
- If the human player wins again, then round3 will start. Thus the game will continue.
- In any stage, if the human player loses, the game will stop. The points achieved by the human player will be saved (appended) to another text file named **Score.txt**.

To implement this Mortal Combat Game, it is highly recommended to use class named as **Player**, **ScoreBoard**, and **Game**. Besides for easiness, student can implement other class, vector, structure and pointer concept.

- At the beginning of the game, the data in **MK.txt** should be read and shown in the output so that human player can choose the **CharacterName**. Then the human player will enter his/her name
- After that the round 1 will start. The point calculation will be done as follows. At the first stage both player will be given a health of 100 points.
- In the second stage they will be given 200 points. In this way, the initial health of every stage will be provided.
- The remaining health of human user will be added to the next level. Whereas, each level starts with new AI player, there will be no carry over for AI player.
- If human players die, his/her health point in the last stage will be considered as point.
- Random number should be used for player turn, type of attack and block/unblock option.

The Sample output will be as followed:

```

                WELL COME TO MORTAL KOMBAT
CharacterName  BlockPower      AttackPower  SpecialPower
Subzero              10              12              30
Scorpion            9              16              25
Raiden              15              20              40
Jaks                 11              13              23
Kitana               13              13              29

```

```
Enter Your Name: Umair
```

```
Which character do you like? Press [1 to 5]:
```

```
=====Stage1 Begins=====
```

```
Umair wins over Subzero with 320 points.
```

```
=====Stage2 Begins=====
```

Scorpion wins over Umair with 110 points.

===== GAME OVER =====

Umair earns 320 points.

Do you want to see all scoreboards [Y/N]: Y

| Name   | Points |
|--------|--------|
| Ragib  | 500    |
| Hassan | 480    |
| Karim  | 310    |
| Umair  | 320    |

**HELP:**

```
const char* cs = s.c_str(); // converts s into the C-string cs
int atoi(const char* s); //Returns the integer represented
literally in the string s.
```

Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 10:

### Snake Ladder Game

Snake Ladder is a classical game involving a board of numbers and players. In this project, you have to implement a customized snake-ladder game following the rules stated below:

#### Rules:

1. A sample Board layout will be like the table below:

|          |           |          |           |           |           |                  |
|----------|-----------|----------|-----------|-----------|-----------|------------------|
| 47       | 46        | 45 (S,9) | 44        | 43        | 42        | Player: Position |
| 36       | 37        | 38       | 39        | 40        | 41        | P1:10            |
| 35(L,38) | 34 (S,22) | 33       | 32 (S,13) | 31        | 30        | P2:16            |
| 29       | 28 (S,16) | 27       | 26 (L,41) | 25 (S,14) | 24        | P3:23            |
| 23       | 22        | 21 (S,7) | 20        | 19        | 18 (L,24) |                  |
| 12 (S,5) | 13 (L,27) | 14       | 15 (S, 7) | 16        | 17        |                  |
| 11       | 10 (L,23) | 9        | 8         | 7         | 6 (L,20)  |                  |
| Start    | 1         | 2        | 3         | 4 (L,17)  | 5         |                  |

**Figure 1:** Initial Layout of the board

2. Two special types of squares: **Snake(S)** and **Ladder (L)**. If any player lands on any of these squares, they have to jump to another position just given in the figure. As for example: 10 (L, 23) means the square of value 10 contains a Ladder and the player's position will jump to 23 from this square. Similarly (S, 9) means the square contains Snake's Head and

the player will jump to Number 9's square from here. The text file **"SnakeLadder.txt"** will keep the position of Snake and Ladder. The board will be initialized with this text file. The file will look like below, where S means Snake and L denotes Ladder. The first number is the initial position whereas the second one is final.

|   |    |    |
|---|----|----|
| S | 30 | 20 |
| L | 15 | 27 |
| S | 17 | 5  |
| L | 34 | 41 |
| S | 35 | 2  |
| L | 5  | 39 |

3. There will be many players. They will roll their dice one by one. The dice has only 3 sides (1, 2, and 3). The outcome of the dice thrown by a player will be chosen randomly.
4. Initially the players will be on Start block. Each player has to drop '1' to start its journey.
5. After each move, the player gets to move to another position based on the conditions shown on the figure of the board.
6. The player who reaches 47 position first will win the match.
7. Remember, if position crosses 47 that will not be allowed.

### Task Achievement:

1. Implement a class named **'Square'** which stores the information of a square in the board, such as: position\_number(int), special\_move(int), move\_to\_position(int). If special\_move=1, then the square contains a snake's head and if special\_move=2, then the square contains the starting point of a ladder. Special move=0 means the square is a normal number.
2. Implement a class named **'Board'** which contains 48 squares (including Start). This class will create this board, display the board just like the layout given on the **Figure-1**
3. Implement a class named **'Player'** which contains information of a player such as name, current\_position, total\_number\_of\_moves etc.
4. Implement a class named **'GamePlay'** which will play the game with necessary options. Initially each player is on the Start square. First Player will be asked to roll a dice. If a player writes 'R', then a number (1,2 or 3) will arrive randomly and that player's position



gets changed according to the rules stated above. Next the second player will be asked to roll a dice and the process continues in round-robin fashion. After each step print each player's position number and possibly display the board, indicating each player's position on the console.

5. Implement a class named '**ScoreBoard**' which will show the name of the player, matches played by him and the number of winning matches. This data should be stored in a file named "GameScore.txt". This file must be updated before leaving the game.

| PlayerName | Number_Playing | Number_Winning |
|------------|----------------|----------------|
| Raihan     | 10             | 7              |
| Rakeeb     | 20             | 9              |
| Razeen     | 25             | 15             |
| Rishat     | 15             | 10             |
| Riaz       | 13             | 6              |

**Note:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 11

### Hotel Management System

Write a C++ program to implement a hotel management system which has the features as below:

1. There will be 3 types of user mode: **Manager, Customer and Owner**.
2. To enter as Manager, user need to provide a password. Password is: '#manager@223'. Manager can view the booking requests, assign rooms to the customer as per the request, view booked guests' details, checkout room of existing guest and calculate bill of the customer after check out.
3. Customer mode does not need any password to enter. Customers can view the available rooms and place any booking request. A customer can place a booking request of only 1 room. During placing the request, the system will ask to enter the name of the customer and requested room number.
4. To enter as Owner, user need to provide a password. Password is: '#admin@123'. Owner can view the total number of rooms booked in a month, net expenditure of the hotel in a month and net profit of the month. Owner can input the information of employ, staff salary and other expenditure of the hotel for a specific month.
5. Assume that there are 50 rooms in the hotel. The hotel has been divided into 5 blocks. The block allotment and room cost have been given below:

**Table-1**

| Room No | Block no | Room Cost (per day) |
|---------|----------|---------------------|
| 01-10   | A        | 25\$                |
| 11-25   | B        | 10\$                |
| 26-40   | C        | 15\$                |
| 41-50   | D        | 20\$                |

6. **Room Service:** Manager can add room service employees to different blocks. There are 2 shifts of work: Day (8 am - 6pm) and Night (7pm - 8am). While adding employees, the

Manager has to input the name of the employee, an ID, assign a block and a shift. Customers can request room service to manager with the specified time in string, manager will see the available room service employees of that block and assign an employee to that room. If no employee of that block is available, Manager can search and assign employees from other blocks also. Manager will update the employer's status and store room service information of the room after the completion of room service. Manager can also update the information of total expenditure behind hotel rooms

7. **Bill Information:** Cost of rooms have been given on the Table-1 . We assume customers will stay for one day only. During check out, the manager will calculate the bill with additional room service cost (if any). Room service will add \$5 extra to the bill upon calling every time.
8. **Hotel Sales:** We assume all rooms of Block-A and Block-C were booked for the whole month of September by different customers. Total expenditure for the month of September is as follows: behind each booked rooms cost was 5\$ per day, for vacant rooms cost was 2\$ per day. Employee and staff salary, and other expenditure is calculated by the owner.

### Task Achievements:

1. Implement a class called '**Manager**' which will have the following functions:
  - `void viewBookingRequest() ,`
  - `void assignRooms(string customerName, int roomNumber) ,`
  - `void viewBookedRoomDetails() ,`
  - `void checkOut(string customerName) ,`
  - `void addRoomServiceEmployee(int employeeID, char blockNo, int shift) ,`
  - `void assignRoomService(int employeeID, int roomNumber) ,`
  - `void endRoomService(int employeeID, int roomNumber) ,`
  - `int roomsBooked(int roomCosts) ,`
  - `int calculateBill(int roomNumber) ,`
  - `int roomsBookedMonth(int roomsMonth) ,`
  - `int calculateRoomsMonth (int bookedRoomcost, int vacantRoomcost) .`
2. Implement a class called '**Customer**' which will have the following attributes and methods:
  - `String name ,`

- `void placeRoomBookingRequest(int roomNumber) ,`
- `void showAvailableRoomsWithCost() ,`
- `void placeRoomServiceRequest() .`

3. Implement a class called **‘Owner’** which will have the following functions:

- `int OtherExpenditure (int employeeSalary, int staffSalary, intOtherCost) ,`
- `void roomsBookedMonth() ,`
- `int netExpenditure(int calculateRoomsMonth, int OtherExpenditure) ,`
- `int netProfit(int roomsBooked, int netExpenditure) .`

4. Implement a class called **‘HotelManagement’** to show necessary input options after running the code and handle the task. First you will be asked to enter either manager mode or customer mode. If you enter manager mode, then manager’s work menu will be shown, then you can choose any of the manager’s work to do with necessary input options. Same goes for customer Mode. You should also have a logout option. This class will store the summary of room bookings and room service in a ‘room.txt’ file.
5. Implement a class called **‘RoomServiceEmployee’** which stores the information about the room service employee.
6. You can also create your own class and methods as per your requirements.

#### Note:

4. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
5. Students are not allowed to use the concepts beyond the syllabus.
6. Programming language is C++. Try to avoid the coding pattern of C.

## **Project 12**

### **Library Management System**

In this project, implement a library management system according to the following features:

1. Two types of users: **Librarian, Student**
2. To enter as Librarian, a user has to provide a username and password. Username is: admin, Password: lib@123
3. A librarian can create an account for students with an ID (int) and password(string). A librarian can
  - add new books to the library,
  - view book issue requests,
  - issue books to the students with a specified Date of Issue,
  - view book return request, return books to the library. While handling book return request admin will check the last date return and actual return date of the book and calculate fine if any.
4. A student can log in to the system with appropriate username and password assigned by the librarian. A student can
  - place a book issue request and
  - book request with a specified date of return.
5. A student can borrow a book only for 1 week. If he/she fails to return the book within the deadline, a fine of 2\$ will be charged against him/her on a per day basis. If a book is severely damaged or lost by a student, he/she has to pay the full price of the book to the library. A student can place a lost or damaged report with the money, and the admin can view the report. If a student places a damaged or lost report 2 times, the admin will delete the account and cancel membership.
6. A deleted member can place a new create account request by paying 20\$ . Admin then creates a new account for that student.

#### **Task Achievements:**

1. Implement a class called '**Librarian**' which will have the following functions:
  - `void createAccount()` ,
  - `void deleteAccount(int studentID)` ,

- `void viewBookIssueRequest() ,`
- `void issueBooks(string bookTitle, int studentID, string issueDate) ,`
- `void viewBookReturnRequest() ,`
- `void returnBook(string bookTitle, string retrunDate, int studentID)`

2. Implement a class called '**Student**' which will have the following attributes and methods:

- `int studentID,`
- `string password.`
- `void placeIssueRequest(string bookTitle) ,`
- `void BookReturnRequest(string returnDate) ,`
- `void placeLostDamageRequest()`

3. Implement a class called '**Book**' to store the information about book like:

- `string title,`
- `int price,borrowStatus,`
- `int lastDateofReturn,`
- `int borrowedStudentID`

4. Implement a class called '**Library**' to create necessary user options, menus after running the code. After running the code, users can log in as admin or students. Then the list of tasks will be shown to the user according to user type. After choosing a task, the code must call necessary objects and methods to handle the task. There should be a log out option also.

5. A summary report containing the book's issue history, student's borrow and return history and total funds received by the library should be stored in file called "**library.txt**".

6. You can define more attributes or methods to utilize the above mentioned classes to achieve any feature goal.

#### **Note:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## **Project-13**

### **Book Shop Management System**

Write a C++ program to maintain the activities of a book shop that includes the following features:

- i. There should be a login option for two types of users- admin and customer.
- ii. Admin will be able to register, modify, delete and view book details, i.e., title, name of the author, publisher, edition, price, and stock position. Default admin username is: “admin”, and password is: “12345678”.
- iii. Customers need to login with a valid username and password. Customers can search for their desired book by providing book title and author’s name. If the book isn’t available, system will show an appropriate message. If it is available, system will display the details of that book and will request for the number of copies required. Customers can either buy or borrow books. In order to borrow books, one need to pay 20% of the original price and must return it within 7 days. If the required copies are available price will be displayed, otherwise “stock out” will be displayed.
- iv. Admin can see the information of customers, i.e., provided personal data and order details.

#### **Task Achievement:**

To implement this program the user should follow the following steps:

- Implement a class named **admin** which will have the functions: **newBook ()**, **editBook ()**, **deleteBook ()**, **viewBook ()**, and **seeCustomerInfo ()**.
- Implement a class named **customer** which will have a customer ID number for each customer and keep all the information of the customers (i.e. name, phone no., email id, mailing address, billing address). Use functions **customerInfo ()** and **editInfo ()** to store these data and edit customer information, respectively. Admin can access the information of customers using their customer ID number.
- Implement another class named **menuSelection** which will have functions **searchBook ()**, **viewBook ()**, **borrowBook ()**, and **buyBook ()**.

- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this system.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.
- At the end save all the transaction related to book in a text file named “DailySales.txt”. which looks like below:

| CustomerID | CustomerName | BookID | BookName | Quantity | Status (B/P) | Price |
|------------|--------------|--------|----------|----------|--------------|-------|
| 100200     | Zakir        | 101    | TheLion  | 3        | B            | 60    |
| 100500     | Zahid        | 106    | Kindness | 2        | P            | 500   |
| 100509     | Zaman        | 189    | Eternal  | 1        | P            | 460   |

**Note:**

- Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
- Students are not allowed to use the concepts beyond the syllabus.
- The programming language is C++. Try to avoid the coding pattern of C.
- While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.



## **Project-14**

### Grading Management System for EEE Department

Suppose you are willing to generate an automatic grading system for EEE department. Write a program using C++ that includes the following features:

- i. There should be a login option for two types of users- admin and student.
- ii. Admin will be able to register, modify, delete, view and search student information (i.e. name, student ID, section, year, semester, department). Default admin username is: “admin”, and password is: “austee”.
- iii. Student will be able to search and view his/her information and result only. They need to login with a valid username and password.

#### **Task Achievement:**

To implement this program the user should follow the following steps:

- Implement a class named **courses** that will keep information of different courses that are offered in a semester. There should be 8 semesters and under each semester there will be different courses (theory and sessional). Get help from “calendar 2013” to get information about different courses.
- Implement a class named **studentInfo** that will keep all the information of student. Use functions **newData ()**, **editData ()**, **viewData ()**, **deleteData ()**, and **searchData ()**. Admin will be able to view all information of the students. Admin can search for a student by providing his/her student ID only.
- After providing student’s year-semester information, courses under that particular semester should be displayed only. Admin will provide marks of respective courses and GPA will be calculated automatically.
- Implement a class named **result** that will have functions **calculateResult ()** and **viewResult ()**.

- Implement a class named **student**, using which a student can see his/her information and result using **searchData ()** and **viewResult ()** functions, respectively.
- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this system.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.
- Save the results in a txt file named as “**grading.txt**” like below:

| StudentID | StudentName | SubjectCode | Year | Semester | Grade |
|-----------|-------------|-------------|------|----------|-------|
| 170501333 | Arif        | EEE2109     | 1996 | Fall     | A     |
| 170501333 | Arif        | EEE2110     | 1996 | Fall     | A     |
| 170501333 | Arif        | EEE2111     | 1996 | Fall     | A     |
| 170501333 | Arif        | EEE2112     | 1996 | Fall     | B     |
| 170501337 | Zahid       | EEE2113     | 2000 | Spring   | A     |
| 170501338 | Jaheer      | EEE2114     | 2001 | Spring   | A     |
| 170501339 | Asif        | EEE2115     | 2002 | Spring   | C     |

**Note:**

- Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
- Students are not allowed to use the concepts beyond the syllabus.
- The programming language is C++. Try to avoid the coding pattern of C.
- While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.

## **Project 15.**

### **StockMarket**

Write a program to help a local stock trading company automate its systems. The company invests only in the stock market. At the end of each trading day, the company would like to generate and post the listing of its stocks so that investors can see how their holdings performed that day. We assume that the company invests in, say, 10 different stocks. The desired output is to produce two listings,

- one sorted by stock symbol [alphabetic order]
- another sorted by percent gain from highest to lowest.

The input data is provided in a file in the following format:

| symbol | openingPrice | closingPrice | todayHigh | todayLow | prevClose | volume  |
|--------|--------------|--------------|-----------|----------|-----------|---------|
| MSMT   | 112.5        | 115.75       | 116.5     | 111.75   | 113.5     | 6723823 |
| CBA    | 67.5         | 75.5         | 78.75     | 67.5     | 65.75     | 378233  |

### **Design and implement a stock object:**

Design a class that captures the various characteristic of a stock object **stockType**. The main components of a stock are the **stock\_symbol**, **stock\_price**, and **number\_of\_shares**. Moreover, we need to output the opening price, closing price, high price, low price, previous price, and the percent gain/loss for the day. These are also all the characteristics of a stock. Therefore, the stock object should store all this information. Perform the following operations on each stock object:

- i. Set the stock information[Either create new data and/or read “**StockMarket.txt**”]
- ii. Print the stock information.
- iii. Show the different prices.
- iv. Calculate and print the percent gain/loss.  $[(\text{closingPrice} - \text{previousPrice}) / \text{previousPrice}] \times 100\%$
- v. Show the number of shares.

## **Project 16.**

### Date System

Design the class **dateType** to implement the date in a program. The member function **setDate** and the constructor should check whether the date is valid before storing the date in the member variables. Definitions of the function **setDate** and the constructor would be such that the values for the month, day, and year are checked before storing the date into the member variables. Add a member function, **isLeapYear**, to check whether a year is a leap year.

The above mentioned class **dateType** was designed and implemented to keep track of a date, but it has very limited operations. Redefine the class **dateType** so that it can perform the following operations on a date, in addition to the operations already defined:

- a. Set the month.
- b. Set the day.
- c. Set the year.
- d. Return the month.
- e. Return the day.
- f. Return the year.
- g. Test whether the year is a leap year.
- h. Return the number of days in the month. For example, if the date is 3-12-2013, the number of days to be returned is 31 because there are 31 days in March.
- i. Return the number of days passed in the year. For example, if the date is 3-18-2013, the number of days passed in the year is 77. Note that the number of days returned also includes the current day.
- j. Return the number of days remaining in the year. For example, if the date is 3-18-2013, the number of days remaining in the year is 288.
- k. Calculate the new date by adding a fixed number of days to the date. For example, if the date is 3-18-2013 and the days to be added are 25, the new date is 4-12-2013

The class **dateType** defined in the above task prints the date in numerical form. Some applications might require the date to be printed in another form, such as March 24, 2013. Derive the class **extDateType** so that

- a. the date can be printed in either form.
- b. Add a member variable to the class **extDateType** so that the month can also be stored in string form.
- c. Add a member function to output the month in the string format, followed by the year—for example, in the form March 2013.

Write the definitions of the functions to implement the operations for the class **extDateType**.

Design and implement a class **dayType** that implements the day of the week in a program. The class **dayType** should store the day, such as Sun for Sunday. The program should be able to perform the following operations on an object of type **dayType**:

- a. Set the day.
- b. Print the day
- c. Return the day.
- d. Return the next day.
- e. Return the previous day.
- f. Calculate and return the day by adding certain days to the current day. For example, if the current day is Monday and we add 4 days, the day to be returned is Friday. Similarly, if today is Tuesday and we add 13 days, the day to be returned is Monday.
- g. Add the appropriate constructors.

Using the classes **extDateType** and **dayType** the class **calendarType** so that, given the month and the year, we can print the calendar for that month. To print a monthly calendar, you must know the first day of the month and the number of days in that month. Thus, you must store the first day of the month, which is of the form **dayType**, and the month and the year of the calendar. Clearly, the month and the year can be stored in an object of the form **extDateType** by setting the day component of the date to 1 and the month and year as specified by the user. Thus, the class **calendarType** has two member variables: an object of the type **dayType** and an object of the type **extDateType**.

Design the class **calendarType** so that the program can print a calendar for any month starting January 1, 1500. Note that the day for January 1 of the year 1500 is a Monday. To calculate the first day of a month, you can add the appropriate days to Monday of January 1, 1500.

For the class **calendarType**, include the following operations:

- a. Determine the first day of the month for which the calendar will be printed. Call this operation **firstDayOfMonth**.
- b. Set the month.
- c. Set the year.
- d. Return the month.
- e. Return the year.
- f. Print the calendar for the particular month.
- g. Add the appropriate constructors to initialize the member variables.

Design the class **calendarType** so that the program can print a calendar for any month starting January 1, 1500. Note that the day for January 1 of the year 1500 is a Monday. To calculate the first day of a month, you can add the appropriate days to Monday of January 1, 1500. For the class **calendarType**, include the following operations:

- a. Determine the first day of the month for which the calendar will be printed. Call this operation **firstDayOfMonth**.
- b. Set the month.
- c. Set the year.
- d. Return the month.
- e. Return the year.
- f. Print the calendar for the particular month.
- g. Add the appropriate constructors to initialize the member variables.

Write a test program to print the calendar for either a particular month or a particular year. For example, the calendar for September 2013 is: September 2013

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|-----|-----|-----|-----|-----|-----|-----|
| 1   | 2   | 3   | 4   | 5   | 6   | 7   |
| 8   | 9   | 10  | 11  | 12  | 13  | 14  |
| 15  | 16  | 17  | 18  | 19  | 20  | 21  |
| 22  | 23  | 24  | 25  | 26  | 27  | 28  |
| 29  | 30  |     |     |     |     |     |

## **Project 17.**

### TRAIN

Define a class named TRAIN. Include following members:

Data members :-

- Train Number
- Name of the train
- Starting station
- Arrival station
- Departure time
- Arrival time
- Off-day

Member functions

- To initialize members
- To input train details
- To display data

Once you have written the class, write a program that creates four TRAIN objects and stores the following data in them by reading the file “train.txt”



| TRAIN NO | NAME OF THE TRAIN | OFF-DAY | STARTING STATION | DEPARTURE TIME | ARRIVAL STATION | ARRIVAL TIME |
|----------|-------------------|---------|------------------|----------------|-----------------|--------------|
| 702      | SUBARNA EXPRESS   | FRIDAY  | DHAKA            | 16:30          | CHATTOGRAM      | 21:50        |
| 704      | MAHANAGAR PROVATI | NO      | DHAKA            | 07:45          | CHATTOGRAM      | 14:00        |
| 705      | EKOTA EXPRESS     | TUESDAY | DHAKA            | 10:10          | DINAJPUR        | 21:00        |
| 709      | PARABAT EXPRESS   | TUESDAY | DHAKA            | 06:20          | SYLHET          | 13:00        |

For better management of the program create a vector of objects of class **Train**.

These objects can be visible in a list as shown above by both of the class **StationMaster** and **Passenger**.

**StationMaster** can do the following things:

- Set,edit,delete any train schedule
- See any train schedule

**Passenger** can do the following things:

- See any train schedule
- Purchase ticket for available seats.

## **Project 18**

### **Canteen Management System**

Develop an object-oriented program in C++ to create a simple Canteen Management System (CMS) with the following information read from the keyboard:

**Please, Choose Your Option:**

1. Add New Sale
2. View All Items
3. Sold Items
4. Add New Item
5. Edit Item
6. Delete Item
7. Total Sales Report

Enter Your Choice: \_

Now with choice enter the following information's.

- Add New Sale –  
Invoice ID (Auto-serial no) + Date + Customer Name + Items + Items quantity. (check the 'Items quantity' with the inventory/all item list)
- View All items, Sold Items, Add New Item, Edit Item –  
Item Id + Item name + Item quantity + Item unit price
- Delete Item –  
Take Input - item ID or Item name + Delete item from inventory
- Total Sales Report –  
Display total sales details

#### **Task Achievement:**

To implement the program the user should follow the following steps:

- Implement a class named **AllItems** which will have all the information about all Items (Item Id + Item name + Item quantity + Item unit price) and have the functions : **AddNewItem()**, **EditItem()** and **DeleteItem()**
- Implement a class named **SoldItems** which is used to update the information of sold items (Item Id + Item name + Item quantity + Item unit price + Total amount) and have the function:  
**TotalSalesReport()**.
- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this program.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.
- Save the total sales report in a text file named as “**canteen.txt**”.

Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## **Project 19**

### Travel Agency Management System

Develop an object-oriented program in C++ to create a simple Travel Agency Management System (TAMS) with the following information read from the keyboard:

**Please, Choose Your Option:**

1. **New Travel Trip.**
2. **Show All Users.**
3. **Edit User.**
4. **Delete User.**
5. **Add Trip.**
6. **Edit Trip.**
7. **Delete Trip.**

Enter Your Choice: \_

Now with choice enter the following information's.

- **New Travel Trip –**  
Invoice ID (Auto-serial no) + Enter Date + Enter User Name + Enter User Address + Enter User Phone No. (11 digit) + Enter User Email Address + Enter Trip Date + Enter Trip Location (Start location and Final Destination)
- **Show All Users, Delete User, Edit User –**  
Enter Invoice ID + Enter Date + Enter User Name + Enter User Address + Enter User Phone No. (11 digit) + Enter User Email Address + Enter Trip Date + Enter Trip Location (Start location and Final Destination) [Modify as require]
- **Add Trip, Edit Trip, Delete Trip –**  
Enter Invoice ID + Enter Trip Date + Enter Trip Location (Start location and Final Destination) [Modify as require]

### Task Achievement:

To implement the program the user should follow the following steps:

- Implement a class named **NewTravelTrip** which will have all the information about (Invoice ID + Enter Date + Enter Trip Location (Start location and Final Destination) ) and have the functions : **AddTrip()**, **EditTrip()** and **Delete Trip()**.
- Implement a class named **ShowAllUsers** which is used to update the information of all users information as (Enter User Name + Enter User Address + Enter User Phone No. (11 digit) + Enter User Email Address) and have the functions: **DeleteUser()** and **EditUser()**.
- Write necessary attributes and methods to utilize the above-mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this program.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.
- Save the active trip details in a text file named “**travel.txt**”.

### Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 20

### Pharmacy Management System

Develop an object-oriented program in C++ to create a simple Pharmacy Management System (PMS) with the following information read from the keyboard:

**Please, Choose Your Option:**

1. Add New Sale
2. View All Medicine
3. Sold Medicine List
4. Add New Medicine
5. Update Medicine Details
6. Delete Medicine From Stock
7. Total Sales Report

Enter Your Choice: \_

Now with choice enter the following information's.

- Add New Sale –  
Invoice ID (Auto-serial no) + Date + Customer Name + Medicine Name +  
Company Name (Auto from the list) + Medicine quantity (check the 'Items  
quantity' with the inventory/all item list) + Medicine price (Auto from the  
list) + Show Total Amount.
- View All Medicine, Sold Medicine List, Add New Medicine, Update Medicine Details –  
Medicine Name + Company Name + Medicine Quantity + Medicine Unit  
price + Medicine Arrival Date + Medicine Expire Date
- Delete Medicine From Stock –  
Take Input - Medicine Name + Delete Medicine From Stock

#### Task Achievement:

To implement the program the user should follow the following steps:

- Implement a class named **AddNewSale** which will have all the information about the medicine sale (Invoice ID (Auto-serial no) + Date + Customer Name + Medicine Name) and have the functions : **AddNewMedicine ()**, **UpdateMedicineDetails()** and **DeleteMedicineFromStock ()**
- Implement a class named **SoldMedicineList** which is used to update the information of sold medicine (Medicine Name + Company Name +Medicine Quantity + Medicine Unit price + Medicine Arrival Date + Medicine Expire Date) and have the functions: and **TotalSalesReport()**.
- Write necessary attributes and methods to utilize the above mentioned classes in logical way. Use necessary getter and setter methods with menu options for the ease of the operation of this program.
- Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.
- Save the total sales report in a text file named as “**pharmacy.txt**”.

Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C

## **Project 21**

### **Pay Roll System**

Develop an object-oriented program in C++ to create a **pay roll system** of an organization with the following information read from the keyboard:

*Employee details - First name, Last name, Year, Department, Email, Phone, Address*

*Employee code-Alphanumeric Code*

*Designation- Teachers(Professor, Associate, Assistant), Officers( Head of Account, Librarian, Registrar)*

*Account number- 11-digit number*

*Date of joining- dd/mm/yy format*

*Salary-Basic+ House rent+ Medical Allowance+ Technical support (Not for officers)*

*Dearness Allowance- 5% of the Basic*

*Festival Bonus-1 Basic twice*

*Providant Fund- 5% of the Basic*

*Tax dedeuction- 5% of the Basic*

#### **Task Achievement:**

To implement the program the user should follow the following steps:

Design a base class consisting of *employee name*, *employee code* and *designation* and another base class consisting of the data member, such as *account number* and *date of joining*. The derived class consists of the data member of *basic pay* plus *other earnings and deductions*.



The program should have the facilities.

- a) Build a master table
- b) List a table
- c) Insert a new entry
- d) Delete old entry
- e) Edit an entry
- f) Search for a record.

Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## Project 22

### Hospital Management System

Develop an object-oriented program in C++ to create a simple **Hospital Management System (HMS)** with the following information read from the keyboard:

There will be a prompt message with login password at the begin of the system

```
HOSPITAL MANAGEMENT SYSTEM
-----
LOGIN
-----
Enter Password:
```

Use 'pass' as password and keep a system so that the password can be modified later on. Inaccurate password would show maximum 2 more attempts and then abort the program. After a successful login the system would show following screen

```
HOSPITAL MANAGEMENT SYSTEM

Please, Choose from the following Options:

1 >> Add New Patient Record
2 >> Add Diagnosis Information
3 >> Full History of the Patient
4 >> Information About the Hospital
5 >> Exit the Program

Enter your choice:
```

Now with choice enter the following information's.

*Add New Patient Record-Name. Address, Contact Number, Age, Gender, Blood Group, Earlier history (100 words), Patient ID*

*Medical Information (100 words)-Symptoms, Diagnosis, Medicines, Admission required, Type of Wards*

*Full History of the Patient- New patient record+ Medical Information*

*Information about hospital-A welcome speech from the Hospital management*

**Task Achievement:**

To implement the program the user should follow the following steps:

Design a base class consisting of *New Patient Record* and *Medical Information* and another base class consisting of the *Information about hospital*. The derived class consists of the data member of *New Patient Record* and *Medical Information*

The program should have the facilities. i) Build a master table ii) List a table iii) Insert a new entry iv) Delete old entry v) Edit an entry vi) Search for a record.

Note:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. Programming language is C++. Try to avoid the coding pattern of C.

## **Project 23**

### **Banking Account System**

Banks offer various types of accounts, such as savings, checking, certificate of deposits, and money market to attract customers as well as meet with their specific needs. Two of the most commonly used accounts are savings and checking. Each of these accounts has various options. For example, you may have a savings account that requires no minimum balance but has a lower interest rate. Similarly, you may have a checking account that limits the number of checks you may write. Another type of account that is used to save money for the long term is certificate of deposit (CD).

#### **PartA:**

Define the class **bankAccount** to store a bank customer's **accountNumber** and **balance**. Suppose that account number is of type **int**, and balance is of type **double**. Your class should, at least, provide the following operations:

- set the account number
- retrieve the account number
- retrieve the balance
- deposit money
- withdraw money
- print account information
- Add appropriate constructors

Every bank offers a checking account. Derive the class **checkingAccount** from the class **bankAccount** (designed in partA). This class inherits members to store the account number and the balance from the base class. A customer with a checking account typically

- receives interest,
- maintains a minimum balance, and
- pays service charges if the balance falls below the minimum balance.

Add member variables to store this additional information. In addition to the operations inherited from the base class, this class should provide the following operations:

- set interest rate,
- retrieve interest rate,
- set minimum balance,
- retrieve minimum balance,
- set service charges,
- retrieve service charges,
- post interest, verify if the balance is less than the minimum balance,
- write a check, withdraw (override the method of the base class),
- print account information.
- Add appropriate constructors.

Every bank offers a savings account. Derive the class **savingsAccount** from the class **bankAccount** (designed in partA). This class inherits members to store the account number and the balance from the base class. A customer with a savings account typically

- receives interest,
- makes deposits, and
- withdraws money.

In addition to the operations inherited from the base class, this class should provide the following operations:

- set interest rate,
- retrieve interest rate,
- post interest, and
- withdraw (override the method of the base class), and
- print account information.
- Add appropriate constructors.

**Savings accounts:** Suppose that the bank offers two types of savings accounts:

- one that has no minimum balance and a lower interest rate and
- another that requires a minimum balance and has a higher interest rate.

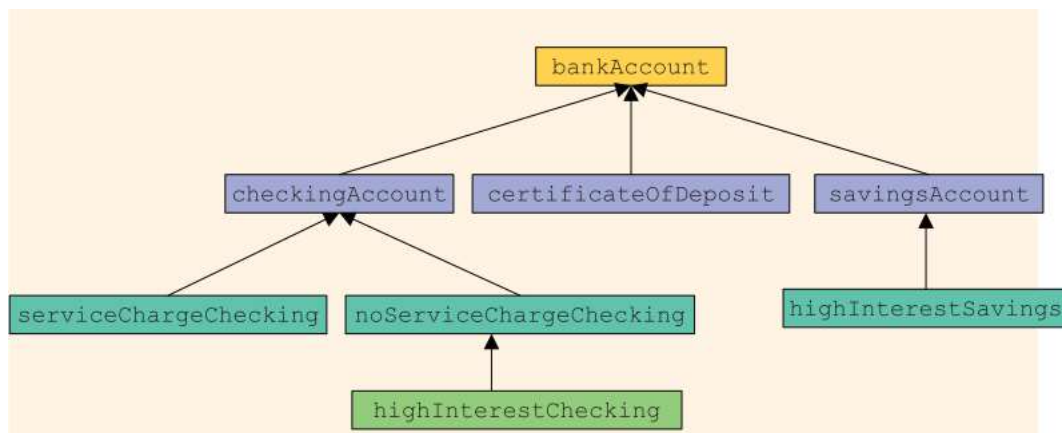
**Checking accounts:** Suppose that the bank offers three types of checking accounts:

- one with a monthly service charge, limited check writing, no minimum balance, and no interest;
- another with no monthly service charge, a minimum balance requirement, unlimited check writing and lower interest; and

- a third with no monthly service charge, a higher minimum requirement, a higher interest rate, and unlimited check writing.

**Certificate of deposit (CD):** In an account of this type, money is left for some time, and these accounts draw higher interest rates than savings or checking accounts. Suppose that you purchase a CD for six months. Then we say that the CD will mature in six months. Penalty for early withdrawal is stiff.

Figure shows the inheritance hierarchy of these bank accounts.



Note that the classes **bankAccount** and **checkingAccount** are abstract. That is, we cannot instantiate objects of these classes. The other classes in Figure are not abstract.

**bankAccount:** Every bank account has an account number, the name of the owner, and a balance. Therefore, instance variables such as name, accountNumber, and balance should be declared in the abstract class bankAccount. Some operations common to all types of accounts are retrieve account owner's name, account number, and account balance; make deposits; withdraw money; and create monthly statement. So include functions to implement these operations. Some of these functions will be pure virtual.

**checkingAccount:** A checking account is a bank account. Therefore, it inherits all the properties of a bank account. Because one of the objectives of a checking account is to be able to write checks, include the pure virtual function writeCheck to write a check.

**serviceChargeChecking:** A service charge checking account is a checking account. Therefore, it inherits all the properties of a checking account. For simplicity, assume that this type of account

does not pay any interest, allows the account holder to write a limited number of checks each month, and does not require any minimum balance. Include appropriate named constants, instance variables, and functions in this class.

**noServiceChargeChecking:** A checking account with no monthly service charge is a checking account. Therefore, it inherits all the properties of a checking account. Furthermore, this type of account pays interest, allows the account holder to write checks, and requires a minimum balance.

**highInterestChecking:** A checking account with high interest is a checking account with no monthly service charge. Therefore, it inherits all the properties of a no service charge checking account. Furthermore, this type of account pays higher interest and requires a higher minimum balance than the no service charge checking account.

**savingsAccount:** A savings account is a bank account. Therefore, it inherits all the properties of a bank account. Furthermore, a savings account also pays interest.

**highInterestSavings:** A high-interest savings account is a savings account. Therefore, it inherits all the properties of a savings account. It also requires a minimum balance.

**certificateOfDeposit:** A certificate of deposit account is a bank account. Therefore, it inherits all the properties of a bank account. In addition, it has instance variables to store the number of CD maturity months, interest rate, and the current CD month.

**Task:**

Write the definitions of the classes described in the above mentioned briefing and write the program to test all of these classes.

## **Project 24.**

### Contact Application

Write a C++ program for the contact app of your phone with the following features.

1. The contact app can have various attributes, such as, **Name**, **Phone number**, **Email**, **Address**, **Date of birth**, etc.
2. There will be login for three types of user account – **owner**, **otherApp** and **guest**.
3. Owner can register or delete other apps as well as guest account, and create username and password for that account. Default Admin username is : "admin" , password is "@dmin@U\$T".
4. Owner can also create/delete/modify any contact attributes such as, Name, Phone number, Email, Address, Date of birth, etc.
5. Other apps (i.e., telegram, signal, whatsapp, viber, etc.) can access/modify the part of the attributes according to the permission.
6. Guest can only view the permitted attributes.

#### **Task Achievement:**

To implement the program the coder should follow the following steps:

1. Implement a class named **Owner** which will have the methods : createContact(), createOtherApp(), createGuest(), etc.
2. Implement a class named **OtherApp** which can function through the methods such as, accessContact(), updateContact(), etc.
3. Implement a class named **Guest** which can view the contact information with viewContact().
4. Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.
5. Save the database in a text file named as "contact.txt"



| Name  | PhoneNumber | Email          | Address    | DOB     |
|-------|-------------|----------------|------------|---------|
| Amin  | 600800900   | amin@aust.edu  | Malibag    | 1011990 |
| Asif  | 600800901   | asif@aust.edu  | Mogbazar   | 1011991 |
| Abir  | 600800902   | abir@aust.edu  | Motijheel  | 1011992 |
| Amina | 600800903   | amina@aust.edu | Mugda      | 1011993 |
| Asifa | 600800904   | asifa@aust.edu | Mohakhali  | 1011994 |
| Abida | 600800905   | abida@aust.edu | MerulBadda | 1011995 |
| Asia  | 600800906   | asia@aust.edu  | Mirpur10   | 1011996 |

### Notes:

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. The programming language is C++. Try to avoid the coding pattern of C.
4. You are free to add appropriate constructors and member functions to initialize, access, and manipulate the data members.
5. While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.

## Project 25.

### Student Fees system

Write a C++ program for managing the Student fees system of AUST with the following features.

1. There will be login for three types of user account – **admin**, **official** and **student**.
2. Admin can register or delete new official as well as student account, and create username and password for that account. Default Admin username is: "admin", password is "@dmin@U\$T".
3. Admin can also create/delete/modify semesters for taking the payment.
4. Officials can update and view the payment history of any students for any department for any semester. They can also create/delete/modify the different fields for payment (i.e., tuition fees, electrical fees, laboratory fees, library fees, etc.)
5. Officials can also modify student's accounts (i.e., name, student ID, year, semester, etc.)
6. By logging in their accounts, students can check their past payment history. Also they can view the detailed payment receipt for each semester.
7. Save the payment data in a text file named "**studentFees.txt**".

| StudentName | StudentID | Year | Semester | Tuition | Laboratory | Library | Others |
|-------------|-----------|------|----------|---------|------------|---------|--------|
| Rahat       | 160502369 | 2017 | Fall     | 28000   | 5000       | 2000    | 3000   |
| Rahat       | 160502369 | 2017 | Spring   | 28000   | 5000       | 2000    | 2000   |
| Rahat       | 160502369 | 2016 | Fall     | 26000   | 5000       | 2000    | 500    |
| Rahat       | 160502369 | 2016 | Spring   | 26000   | 5000       | 2000    | 5000   |
| Rahat       | 160502369 | 2018 | Fall     | 28000   | 5000       | 2000    | 800    |
| Zakir       | 160502260 | 2018 | Spring   | 28000   | 5000       | 2000    | 200    |
| Zakir       | 160502260 | 2018 | Fall     | 28000   | 5000       | 2000    | 200    |
| Zakir       | 160502260 | 2019 | Spring   | 30000   | 5000       | 2000    | 200    |
| Zakir       | 160502260 | 2019 | Fall     | 30000   | 5000       | 2000    | 200    |

### Task Achievement:

To implement the program the coder should follow the following steps:

1. Implement a class named **Admin** which will have the methods: createAccount(), createSemester() and more.
2. Implement a class named **Official** which can function through the methods such as, addPayment(), deletePayment(), updatePayment(), modifyStudent() etc.
3. Implement a class named **Student** which can see his/her past payment history using viewPayment() and payment receipt using viewReceipt().
4. Your code should not be limited to the above classes only. Rather, you can add new classes as your requirement.

**Notes:**

1. Students have complete freedom to use pointer, structure, vector and other concepts to make the code efficient.
2. Students are not allowed to use the concepts beyond the syllabus.
3. The programming language is C++. Try to avoid the coding pattern of C.
4. You are free to add appropriate constructors and member functions to initialize, access, and manipulate the data members.
5. While writing codes keep in mind that, your program must be programmer-friendly as well as user-friendly.