**Assignment-1:**

**Object Oriented Programming (OOP)**

**Session(2023-2027)**

**Due-Date: 24-10-2024**

**Assignment Instructions:**

Each student is required to divide the last two digits of their roll number by 3. Based on the remainder obtained, they will complete the corresponding assignment:

* **Remainder 0 or 2:** Reference materials (raw code) for assignments with remainders 0 and 2 can be found on GitHub: [GitHub Repository](https://github.com/safyanch/).
* **Important Guidelines:**
  + Any form of cheating, plagiarism, or use of AI-generated code (e.g., ChatGPT) will result in zero marks. Violations will be reported to the disciplinary committee for further action.
  + Each student will be required to attend an individual viva (oral examination) to explain their solution and demonstrate understanding.

**==============================================================**

**Reminder-0:**

**E-Mailing System using Object-Oriented Programming (OOP)**

**Objective:**

Develop an object-oriented program that simulates a basic email system, allowing users to create accounts, send and receive emails, and manage their inbox. The program should model common functionalities of popular email services like Gmail, Yahoo, and Hotmail.

**Features:**

1. **Account Management:**
   * Users can create accounts on existing email servers by providing:
     + Username
     + Password
     + Mail Server (e.g., Gmail, Yahoo)
   * The program should include setter and getter methods for managing account information.
2. **Email Operations:**
   * Users can:
     + Compose and send emails to other users.
     + Receive emails from other users.
     + Read messages from their inbox.

**Classes and Attributes:**

1. **Account Class:**
   * Attributes:
     + Username
     + Password
     + Mail Server
   * Methods:
     + Setter and getter methods for each attribute.
2. **MailItem Class:**
   * Attributes:
     + To (Recipient)
     + From (Sender)
     + Title (Subject of the email)
     + Message (Content of the email)
3. **MailServer Class:**
   * The MailServer class will store users' account details and manage their emails.
   * It should handle sending and receiving of emails between accounts, along with authenticating user credentials.

**Main Program:**

The main program should provide a user interface with the following menu options:

1. **Sign In:**
   * Authenticate the user by verifying their username and password.
2. **Sign Out:**
   * Log the user out of the system.
3. **Send Email:**
   * Allow the user to compose and send an email to another account.
4. **Read Inbox Emails (int):**
   * Display a list of all email titles in the inbox.
   * If the user selects an email by entering its index, the system will display the full content of that email.
5. **Exit:**
   * Close the program.

**Inbox Management:**

* The Read Inbox Emails option should provide the following functionalities:
  + **List Email Titles:** Show the titles of all emails in the inbox.
  + **Select and Read Email:** When the user inputs the index of an email, display the content of that specific email.
  + **Return to Menu:** After reading an email, return to the main menu for further actions.

**========================== =================**

Reminder**-1**

**BSCS Admission Program for the Department of Software Engineering**

**Objective:**

Develop a program to manage the BSCS admission process, incorporating eligibility and merit criteria for the Software Engineering department.

**Admission Criteria:**

The final score for admission will be calculated based on the following weightage:

* **Matric Marks**: 10%
* **F.Sc Marks**: 60%
* **Test Marks**: 30%
* **Interview Marks**: 10%

**Eligibility Criteria:**

To be eligible for admission to the BSCS program, a student must have at least **50% marks in F.Sc**.

**Program Requirements:**

The program should facilitate the following functionalities:

1. **Form Entry**:
   * Students will be required to fill out an admission form, which includes the following mandatory fields:
     + Form Number
     + Name
     + Father's Name
     + Matric Marks
     + F.Sc Marks
     + Test Marks
     + Interview Marks
2. **Eligibility Check**:
   * Based on the eligibility criteria, the program will identify students who qualify for the BSCS admission process.
   * Eligible candidates will be listed for the entry test and assigned a unique roll number.
3. **Test Score Entry**:
   * After the entry test, the scores of each eligible candidate must be entered into the system.
4. **Merit List Generation**:
   * A merit list of the top 200 candidates, based on their overall scores, will be generated.
   * The list should be displayed on the notice board.

**Program Interface:**

The main program should provide the following menu options:

1. **Form Entry** - Input and store the details of students applying for admission.
2. **Test Score Entry** - Enter test scores for eligible candidates.
3. **Display Test List** - Show the list of eligible candidates for the entry test.
4. **Display Final Merit List** - Generate and display the merit list of the top 200 candidates.

**Reminder-2**

**Music Player System**

**Objective:**

Develop a program that simulates a basic music player, allowing users to add songs to the system and play them using an interactive menu.

**Features:**

1. **Add Songs:**
   * Users can enter song details into the music player system.
   * For each song, the user must provide:
     + **Title** - The name of the song.
     + **Path** - The file path of the song on the local computer.
2. **Play Songs:**
   * The system will allow users to select and play songs from the list of added songs.

**Program Requirements:**

1. **Menu Options:** The main program should present the following menu to the user:
   1. **Enter a Song:**
      * Users can input the title and the file path of the song they wish to add to the music player system.
   2. **Run the Music Player:**
      * When this option is selected, the program will:
        + Display a list of all songs that have been added to the system, each with an index number.
        + Prompt the user to select a song by entering its corresponding index number.
        + Play the selected song.

**Example Workflow:**

1. **Start the Program:**
   * Display the following menu:

mathematica

Copy code

1 - Enter a Song

2 - Run the Music Player

3 - Exit

1. **Option 1: Enter a Song**
   * User inputs:
     + Title: "Song A"
     + Path: "C:/Music/SongA.mp3"
   * The song is then stored in the system with the given details.
2. **Option 2: Run the Music Player**
   * Display the list of available songs:

markdown

Copy code

1. Song A

2. Song B

3. Song C

* + If the user selects 1, the program will play "Song A" by accessing its local path.

1. **Option 3: Exit**
   * Close the program.

**Notes:**

* Ensure that the program handles invalid inputs gracefully.
* The song paths should be valid file paths on the user's computer.
* Integrate a method to play audio files based on the path provided.