# Practice Questions

## 1. Books

Create a class `BookAccount` that represents a library book account. The class should have the following **attributes** and **methods**:

**Attributes:**

- `bookID` (**public**): A string representing the unique identifier for the book.
- `borrowerName` (**public**): A string representing the name of the borrower.
- `fine` (**private**): A double representing the fine for overdue days.

**Constructor:**

- A constructor that initializes `bookID` , `borrowerName` , and calculates the `fine` based on the number of overdue days. The fine is calculated as **$0.50** per day.

**Destructor:**

- A destructor that displays a message when the account is closed and prints the attributes of the account.

**Methods:**

- A method `calculateFine(int overdueDays)` that calculates the fine based on the number of overdue days and sets the `fine` attribute.

- A helper method `calculateOverdueDays(int borrowDate, int currentDate)` that calculates the number of overdue days based on the borrow date and current date, represented as integers in the format `YYYYMMDD` . You will need to use this function to get the overdue days.

**Note:** Use the helper function `calculateOverdueDays()` to calculate overdue days, **DO NOT** pass the days directly to `calculateFine()` function. You can take the date in integer format like `YYYYMMDD` or whichever format you prefer. But focus on the logic in this method.

**Ans.:** Here is a C++ program that satisfies all the conditioned mentioned above:

```cpp
#include <iostream>
#include <ctime>
using namespace std;

struct Date
{
    int year;
    int month;
    int date;
};

Date extractDate(int YYYYMMDD)
{
    int year = YYYYMMDD / 10000;
    int month = (YYYYMMDD / 100) % 100;
    int date = YYYYMMDD % 100;

    Date dateObj;
    dateObj.year = year;
    dateObj.month = month;
    dateObj.date = date;
    return dateObj;
}

int getCurrentDate()
{
    time_t tt;
    struct tm *ti;
    time(&tt);
    ti = localtime(&tt);

    int currentDate = 0;
    currentDate += ti->tm_mday;
    currentDate += (ti->tm_mon + 1) * 100;
    currentDate += (1900 + ti->tm_year) * 10000;
    return currentDate;
}

/** ===
 * the following functions were borrowed and modified from
GeeksForGeeks
 * - countLeapYears(Date date)
 * - countPreviousDays(Date dateObj)
 * - getDateDifference(Date start, Date end)
 *
 * @author Abhay Rathi
 * @link https://www.geeksforgeeks.org/find-number-of-days-
between-two-given-dates/
 *  ===
 **/

const int monthDays[12] = {31, 28, 31, 30, 31, 30, 31, 31,
30, 31, 30, 31};

int countLeapYears(Date date)
{
    int years = date.year;

    // check if the given year needs to be considered for
the count of leap years
    if (date.month <= 2)
        years--;

    // we know that a year is a leap year if it is a
multiple of 4, multiple of 400 and not a multiple of 100
    return years / 4 - years / 100 + years / 400;
}
```

```
int countPreviousDays(Date dateObj)
{
    // convert the year to days and add current date
    long int days = dateObj.year * 365 + dateObj.date;

    // convert the month to days
    for (int i = 0; i < dateObj.month - 1; i++)
        days += monthDays[i];

    // add all leap years before the current year
    days += countLeapYears(dateObj);
    return days;
}

int getDateDifference(Date start, Date end)
{
    int pStart = countPreviousDays(start);
    int pEnd = countPreviousDays(end);
    return pEnd - pStart;
}

// === relevant portion ===
class BookAccount
{
    double fine;
    float fineMultiplier = 0.50;

    int calculateOverdueDays(int borrowDate, int
currentDate)
    {
        Date start = extractDate(borrowDate);
        Date end = extractDate(currentDate);

        int overdue = getDateDifference(start, end);
        return overdue;
    }
    double calculateFine(int overdueDays)
    {
        return overdueDays * fineMultiplier;
    }

public:
    string bookID;
    string borrowerName;

    void display()
    {
        cout << "Book ID      : " << bookID << endl;
        cout << "Borrower Name: " << borrowerName << endl;
        cout << "Current Fine : " << fine << "$" << endl;
    }

    BookAccount(string bID, string bName, int bDate)
    {
        bookID = bID;
        borrowerName = bName;

        int overdueDays = calculateOverdueDays(bDate,
getCurrentDate());
        fine = calculateFine(overdueDays);
    }

    ~BookAccount()
    {
        display();
        cout << "Account closed." << endl;
    }
};

int main()
{
    BookAccount account("Never Stop Learning", "Shahriar",
20241001);
```

```
    return 0;
}
```

**Output:** The code yields the following output in the Terminal:

```
Book ID      : Never Stop Learning
Borrower Name: Shahriar
Current Fine : 1.5$
Account closed.
```

## 2. Room

Design a class `Room` with **private** attributes `length`, `width`, `height`.

- The **constructor** should set the values of these attributes either by user input or with default dimensions (`length = 12`, `width = 8`, `height = 10`)
- Create `Room` objects: one with default dimensions, one with user input, one with passing parameters directly from code.
- Try creating an **object pointer** that would point to one of the existing objects.
- Try creating **new objects with pointers** (with default values, with parameterized values, **with copied values from an existing one**)
- Implement a method to calculate and display the floor area of the room (`length * width`)
- Implement a **non-member method** which will take two `Room` objects as parameters and return the room with larger volume (You can do this in two ways, either using friend function or using using getter methods)

**Ans.:** Here is my attempt to create a C++ program that satisfies all of the conditions above:

```cpp
#include <iostream>
using namespace std;

class Room
{
    double length, width, height;

public:
    Room()
    {
        length = 12;
        width = 8;
        height = 10;
    }

    Room(string _)
    {
        cout << "Enter Room dimensions:" << endl;
        cin >> length >> width >> height;
    }

    Room(double l, double w, double h)
    {
        length = l;
        width = w;
        height = h;
    }

    Room(const Room &room)
    {
        length = room.length;
        width = room.width;
        height = room.height;
        cout << "Copy constructor called!" << endl;
    }

    void calcDisplay(int n)
    {
        double area = length * width;
        cout << "Room " << n << " area: ";
        cout << area << " sq units" << endl;
    }

    double getVolume()
    {
        return length * width * height;
    }
};
```

```
Room getLargerRoom(Room r1, Room r2)
{
    if (r1.getVolume() > r2.getVolume())
        return r1;
    return r2;
}

int main()
{
    Room r1;
    Room r2("cin");
    Room r3(24, 32, 48);

    Room *r4 = &r1;
    Room *r5 = &r3;
    Room r6 = r3;

    r1.calcDisplay(1);
    r2.calcDisplay(2);
    r3.calcDisplay(3);
    r4->calcDisplay(4);
    r5->calcDisplay(5);
    r6.calcDisplay(6);

    Room r7 = getLargerRoom(r1, r2);
    r7.calcDisplay(7);
    return 0;
}
```

**Input:**

```
+ Enter Room dimensions:
  48 52 33
```

**Output:** The code yields the following output in the Terminal:

```
Copy constructor called!
Room 1 area: 96 sq units
Room 2 area: 2496 sq units
Room 3 area: 768 sq units
Room 4 area: 96 sq units
Room 5 area: 768 sq units
Room 6 area: 768 sq units
Copy constructor called!
Copy constructor called!
Copy constructor called!
Room 7 area: 2496 sq units
```

## 3. Library

Create a class `LibraryBook` with **private** attributes `bookTitle`, `borrowerName`, and `borrowDays`.

- Create an array of 20 `LibraryBook` objects.
- Write a function to take input details (from user) for each book (book title, borrower name, and days borrowed).
- Take a **2D Array** of 2×2 dimension and try setting the attributes and displaying the outputs.
- Implement a function to display books borrowed for more than a specific number of days (e.g. more than 30 days).
- Write another function to display the total number of books borrowed by a specific borrower.

**Ans.:** Here is my attempt to create a C++ program that satisfies all of the conditions above:

```cpp
#include <iostream>
#define maxBooks 20
#define maxBorrowDays 30
#define matrixSize 2
using namespace std;

class LibraryBook
{
    string bookTitle;
    string borrowerName;
    int borrowDays;

public:
    LibraryBook()
    {
        bookTitle = "Unknown";
        borrowerName = "Unknown";
        borrowDays = 0;
    }

    LibraryBook(string t, string n, int d)
    {
        bookTitle = t;
        borrowerName = n;
        borrowDays = d;
    }

    void setDetails(string t, string n, int d)
    {
        bookTitle = t;
        borrowerName = n;
        borrowDays = d;
    }

    string getBorrowerName()
    {
        return borrowerName;
    }

    string getTitle()
    {
        return bookTitle;
    }

    int getBorrowDays()
    {
        return borrowDays;
    }

    void getDetails(int n)
    {
        cout << "Book " << n << ":" << endl;
        cout << "Book Title : " << bookTitle << endl;
        cout << "Borrower   : " << borrowerName << endl;
        cout << "Borrow Days: " << borrowDays << endl;
        cout << endl;
    }
};

void getInputFromUser(LibraryBook books[maxBooks])
{
    for (int i = 0; i < maxBooks; i++)
    {
        LibraryBook book;

        string title, borrower;
        int borrowDays;
        cout << "+ Enter details for Book " << (i + 1) <<
": " << endl;
        cin >> title;
        cin >> borrower;
        cin >> borrowDays;
        cout << endl;
        book.setDetails(title, borrower, borrowDays);

        books[i] = book;
    }
}

void displayNewBooks(LibraryBook newBooks[matrixSize]
[matrixSize])
{
    cout << endl;
    cout << "Here are the newly borrowed books:" << endl;
    for (int i = 0; i < matrixSize; i++)
        for (int j = 0; j < matrixSize; j++)
            newBooks[i][j].getDetails(1 + (matrixSize * i +
j));
}
```

```cpp
void displayDueBooks(LibraryBook books[maxBooks])
{
    cout << endl;
    cout << "These books were borrowed more than " <<
maxBorrowDays << " days ago:" << endl;
    for (int i = 0; i < maxBooks; i++)
        if (books[i].getBorrowDays() > maxBorrowDays)
            cout << " - " << books[i].getTitle() << endl;
}

void getBorrowedBooksByBorrower(LibraryBook
books[maxBooks])
{
    string borrower;
    cout << endl;
    cout << "Enter a borrower's name: " << endl;
    cin >> borrower;

    int n = 0;
    for (int i = 0; i < maxBooks; i++)
        if (books[i].getBorrowerName() == borrower)
            n++;
    cout << n << " books were borrowed by " << borrower <<
"." << endl;
}

int main()
{
    LibraryBook books[maxBooks] = {};
    getInputFromUser(books);
    displayDueBooks(books);
    getBorrowedBooksByBorrower(books);

    LibraryBook newBooks[matrixSize][matrixSize] = {
        {LibraryBook("Never Stop Learning", "Shahriar", 7),
LibraryBook("How to Talk to Anyone", "Abrar", 7)},
        {LibraryBook("Atomic Habits", "Fikrat", 35),
LibraryBook("The Art of Communication", "Arefin", 14)}};
```

```
    displayNewBooks(newBooks);

    return 0;
}
```

**Input:**

```
+ Enter details for Book 1:
  Book1 Shahriar 32

+ Enter details for Book 2:
  Book2 Shahriar 48

+ Enter details for Book 3:
  Book3 Shahriar 22

+ Enter details for Book 4:
  Book4 Shahriar 12

+ Enter details for Book 5:
  Book5 Redowan 32

+ Enter details for Book 6:
  Book6 Redowan 14

+ Enter details for Book 7:
  Book7 Manzirul 14

+ Enter details for Book 8:
  Book8 Manzirul 15

+ Enter details for Book 9:
  Book9 Manzirul 16

+ Enter details for Book 10:
  Book10 Rahul 11
```

```
+ Enter details for Book 11:
  Book11 Rahul 7

+ Enter details for Book 12:
  Book12 Protiva 7

+ Enter details for Book 13:
  Book13 Protiva 7

+ Enter details for Book 14:
  Book14 Surayea 14

+ Enter details for Book 15:
  Book15 Ema 14

+ Enter details for Book 16:
  Book16 Fatema 14

+ Enter details for Book 17:
  Book17 Fatema 14

+ Enter details for Book 18:
  Book18 Ismail 14

+ Enter details for Book 19:
  Book19 Ismail 7

+ Enter details for Book 20:
  Book20 Ismail 7
```

**Output:**

```
These books were borrowed more than 30 days ago:
  - Book1
  - Book2
  - Book5
```

**Input:**

```
+ Enter a borrower's name:
  Shahriar
```

**Output:**

```
4 books were borrowed by Shahriar.

Here are the newly borrowed books:
Book 1:
Book Title : Never Stop Learning
Borrower   : Shahriar
Borrow Days: 7

Book 2:
Book Title : How to Talk to Anyone
Borrower   : Abrar
Borrow Days: 7

Book 3:
Book Title : Atomic Habits
Borrower   : Fikrat
Borrow Days: 35

Book 4:
Book Title : The Art of Communication
Borrower   : Arefin
Borrow Days: 14
```

## 4. Streaming Platfrom

- You are creating a system where a streaming platform can access and display a **private** rating for a movie.
- The `Movie` class contains **private** attributes `title` (**string**) and `rating` (**float**), while the `StreamingPlatform` class has a method `displayRating()` that is a **friend** of the `Movie` class. This method allows the platform to access and display the **private** `rating` of the movie.

**Ans.:** Here is my attempt to create a C++ program that satisfies all of the conditions above:

```
#include <iostream>
using namespace std;

class Movie;
class StreamingPlatform
{
public:
    void displayRating(Movie movie);
};

class Movie
{
    string title;
    float rating;
    friend void StreamingPlatform::displayRating(Movie
movie);

public:
    Movie(string t, float r)
    {
        title = t;
        rating = r;
    }
};
```

```
void StreamingPlatform::displayRating(Movie movie)
{
    cout << movie.rating;
}

int main()
{
    Movie featured("The Wild Robot", 4.5);
    StreamingPlatform netflix;
    netflix.displayRating(featured);
    return 0;
}
```

**Output:** The code yields the following output in the Terminal:

```
4.5
```

## 5. Contact Information

Create two classes, `Person` and `Address`. The `Person` class should contain information about a person's **name** and **age**, while the `Address` class should contain details about a person's **address**, including **street**, **city**, and **postal code**.

1. **Classes**
    - Person
        - **Attributes:** `name` (string), `age` (int).
        - **Methods:** Constructor to initialize attributes.
    - Address
        - **Attributes:** `street` (string), `city` (string), `postalCode` (string).
        - **Methods:** Constructor to initialize attributes.

2. **Friend Function**
    - Create a friend function `displayDetails` that takes a Person object and an Address object as parameters and displays the complete details of the person, including their address.

**Ans.:** Here is my attempt to create a C++ program that satisfies all of the conditions above:

```
#include <iostream>
using namespace std;

class Person;
class Address
{
    string street, city, postalCode;
    friend void displayDetails(Person, Address);

public:
```

```cpp
    Address(string s, string c, string p)
    {
        street = s;
        city = c;
        postalCode = p;
    }
};

class Person
{
    string name;
    int age;
    friend void displayDetails(Person, Address);

public:
    Person(string n, int a)
    {
        name = n;
        age = a;
    }
};

void displayDetails(Person person, Address address)
{
    cout << "Name       : " << person.name << endl;
    cout << "Age        : " << person.age << " yrs" << 
endl;
    cout << "Street     : " << address.street << endl;
    cout << "City       : " << address.city << endl;
    cout << "Postal Code: " << address.postalCode << endl;
}

int main()
{
    Person person("Shahriar", 21);
    Address address("Mirpur Thana Road", "Dhaka", "1216");
    displayDetails(person, address);
    return 0;
}
```

**Output:** The code yields the following output in the Terminal:

```
Name       : Shahriar
Age        : 21 yrs
Street     : Mirpur Thana Road
City       : Dhaka
Postal Code: 1216
```