Introductory C++ Tasks: Classes and Member Functions with Solutions

Key C++ Concepts

- **Classes (`class`)**: Define objects with data (fields) and behavior (member functions). Syntax: `class Name { public: type field1; type function(); };`
- **Member Functions**: Functions defined inside a class that operate on its fields.
- **Access Specifiers**: `public` allows access from outside the class.
- **Input/Output**: Use `std::cout` for output, defined in `<iostream>`.
- **Namespaces**: Use `using namespace std;` for simplicity.
- **Libraries**: Include headers like `<string>` for strings, `<cmath>` for math functions.

Task 1: Box Class

Description: Create a class `Box` that stores the length, width, and height (all doubles). Define a member function to calculate the volume (length × width × height). In the `main` function, create a box, set its dimensions, and print the volume.

Solution:

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

class Box {
public:
    double length;
    double width;
    double height;

    double calculateVolume() {
        return length * width * height;
    }
};

int main() {
    Box b;
    b.length = 2.0;
    b.width = 3.0;
    b.height = 4.0;

    cout << "Volume of the box: " << b.calculateVolume() <<
        endl;

    return 0;
}
```

Task 2: Movie Class

Description: Create a class `Movie` that stores the title (string) and rating (double, from 0 to 10). Define a member function to check if the movie is highly rated (rating above 7.0). In the `main` function, create a movie, set its title and rating, and print whether it is highly rated.

Solution:

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

class Movie {
public:
    string title;
    double rating;

    bool isHighlyRated() {
        return rating > 7.0;
    }
};

int main() {
    Movie m;
    m.title = "Inception";
    m.rating = 8.5;

    cout << "Movie: " << m.title << endl;
    if (m.isHighlyRated()) {
        cout << "This movie is highly rated!" << endl;
    } else {
        cout << "This movie is not highly rated." << endl;
    }

    return 0;
}
```

Task 3: Vector2D Class

Description: Create a class `Vector2D` that stores x and y components (both doubles). Define a member function to calculate the magnitude of the vector using the formula $\sqrt{x^2 + y^2}$. In the `main` function, create a vector, set its components, and print the magnitude.

Solution:

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

class Vector2D {
public:
    double x;
    double y;

    double calculateMagnitude() {
        return sqrt(x * x + y * y);
    }
};

int main() {
    Vector2D v;
    v.x = 3.0;
    v.y = 4.0;

    cout << "Magnitude of the vector: " <<
        v.calculateMagnitude() << endl;

    return 0;
}
```

Task 4: BankAccount Class

Description: Create a class `BankAccount` that stores the account holder's name (string) and balance (double). Define a member function to calculate the balance after one year of 5% annual interest (balance $\times$ 1.05). In the `main` function, create an account, set the name and initial balance, and print the balance after one year.

Solution:

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

class BankAccount {
public:
    string holderName;
    double balance;

    double balanceAfterYear() {
        return balance * 1.05;
    }
};

int main() {
    BankAccount acc;
    acc.holderName = "Alice";
    acc.balance = 1000.0;

    cout << "Account holder: " << acc.holderName << endl;
    cout << "Balance after one year: " <<
        acc.balanceAfterYear() << endl;

    return 0;
}
```

Additional Notes

- **Compilation**: Use a C++ compiler (e.g., g++). Example: `g++ filename.cpp -o program` and run with `./program`.
- **Testing**: Verify outputs by changing field values (e.g., different dimensions, ratings, or balances).
- **Debugging**: Check for missing semicolons, incorrect types, or uninitialized variables if errors occur.
- **Extensions**: Add constructors, private fields with getters/setters, or methods for user input.