


The background features a soft, painterly illustration of a mountain range. The mountains are rendered in various shades of light blue and grey, creating a sense of depth and atmosphere. In the upper left corner, three small, dark silhouettes of birds are shown in flight. The overall style is minimalist and serene.

Happy Exam Day!

Recursion

Write a recursive function called `isPalindrome` to check if a given string is a palindrome.

Use the following function declaration: `bool isPalindrome(std::string str)`

The background of the image features a soft, painterly illustration of a mountain range in shades of blue and green. In the upper left corner, three birds are depicted in flight, moving towards the right. The overall aesthetic is calm and scenic.

```
#include <iostream>
#include <string>

bool isPalindrome(std::string str) {
    if (str.length() <= 1) {
        return true;
    }
    else if (str[0] != str[str.length() - 1]) {
        return false;
    }
    else {
        str.erase(str.begin()); // remove first character
        str.erase(str.end() - 1); // remove last character
        return isPalindrome(str);
    }
}
```

Writing Basic Class - setters & getters

write a class definition for a Pet class. This class should have the following properties :

- name (a string to store the pet's name)
- species (a string to specify the kind of animal)
- a setter and getter method for both name and species

Then define both the setters and getters

Defined outside of class

```
class Pet {
    std::string name;
    int age;

    public:
        void setName(std::string name);
        void setAge(int age);

        std::string getName();
        int getAge();
};

void Pet::setName(std::string name) {
    this->name = name;
}

void Pet::setAge(int age) {
    this->age = age;
}

std::string Pet::getName() {
    return name;
}

int Pet::getAge() {
    return age;
}
```

OR

Defined inside of class

```
class Pet {
    std::string name;
    int age;

    public:
        void setName(std::string name) {
            this->name = name;
        }

        void setAge(int age) {
            this->age = age;
        }

        std::string getName() {
            return name;
        }

        int getAge() {
            return age;
        }
};
```

Writing Constructors/Methods

Given this Book class, define the default, parameterized, and copy constructor

* Have the default constructor set title and author to “unknown” and pages and price to 0

```
class Book {
protected:
    std::string title;
    std::string author;
    int pages;
    double price;

public:
    // Default constructor
    Book();

    // Parameterized constructor
    Book(const std::string &title, const std::string &author, int pages, double price);

    // Copy constructor
    Book(const Book &other);
};
```

```
// Default constructor
Book::Book() {
    title = "Unknown";
    author = "Unknown";
    pages = 0;
    price = 0.0;
}


// Parameterized constructor
Book::Book(const std::string &title, const std::string &author, int pages, double price) {
    this->title = title;
    this->author = author;
    this->pages = pages;
    this->price = price;
}

// Copy constructor
Book::Book(const Book &other) {
    title = other.title;
    author = other.author;
    pages = other.pages;
    price = other.price;
}
```


Writing a derived class

create a derived class Ebook that inherits from Book and includes :

- filesize(a double representing the size of the eBook file in megabytes)
- A parameterized constructor that initializes the title, author, pages, price, and fileSize.
- A method displayDetails() that prints out all the details of the eBook including its size



```
class EBook : public Book {
private:
    double fileSize;

public:
    // Parameterized constructor
    EBook(const std::string &title, const std::string &author,
        int pages, double price, double fileSize);

    // Method to display details
    void displayDetails() const;
};
```

```
// Constructor implementation
EBook::EBook(const std::string &title, const std::string &author,
    int pages, double price, double fileSize) {
    this->title = title;
    this->author = author;
    this->pages = pages;
    this->price = price;
    this->fileSize = fileSize;
}

// Method to display details
void EBook::displayDetails() const {
    std::cout << "Title: " << title << std::endl;
    std::cout << "Author: " << author << std::endl;
    std::cout << "Pages: " << pages << std::endl;
    std::cout << "Price: $" << price << std::endl;
    std::cout << "File Size: " << fileSize << " MB" << std::endl;
}
```

In C++, the access specifiers `public`, `private`, and `protected` control the visibility of class members (variables, functions, etc.) and also affect how inheritance works. Here's a brief explanation:

Public Inheritance: When a class is derived from a public base class, the public members of the base class become public in the derived class, and the protected members of the base class become protected in the derived class. Private members of the base class are not accessible directly from the derived class.

Protected Inheritance: When a class is derived from a protected base class, both the public and protected members of the base class become protected in the derived class. Private members of the base class are not accessible directly from the derived class.

Private Inheritance: When a class is derived from a private base class, both the public and protected members of the base class become private in the derived class. Private members of the base class are not accessible directly from the derived class.

In most cases, you'll want to use public inheritance. This is the most common form of inheritance and models an "is-a" relationship (e.g., a Dog is a Mammal). Private and protected inheritance are less common and are typically used to model "is-implemented-in-terms-of" relationships rather than "is-a" relationships.

Constructors/Destructors - What is the Output?

```
#include <iostream>
```

```
class Vehicle {  
public:  
    Vehicle() {  
        std::cout << "This is a vehicle!\n";  
    }  
    ~Vehicle() {  
        std::cout << "Destroying vehicle!\n";  
    }  
};
```

```
class Car : public Vehicle {  
public:  
    Car() {  
        std::cout << "This is a car!\n";  
    }  
    ~Car() {  
        std::cout << "Destroying car!\n";  
    }  
};
```

```
class SportsCar : private Car {  
private:  
    int speed;  
public:  
    SportsCar(int speed) {  
        this->speed = speed;  
        std::cout << "This is a sports car with speed: " << speed << "\n";  
    }  
    ~SportsCar() {  
        std::cout << "Destroying sports car!\n";  
    }  
};
```

```
int main() {  
    Vehicle v;  
    Car c;  
    SportsCar s(200);  
    return 0;  
}
```

Answer:

```
ilannalanton@45 Desktop % ./test
This is a vehicle!
This is a vehicle!
This is a car!
This is a vehicle!
This is a car!
This is a sports car with speed: 200!
Destroying sports car!
Destroying car!
Destroying vehicle!
Destroying car!
Destroying vehicle!
Destroying vehicle!
```

Note: Destructors are called in the reverse order of the constructors. When an object is destroyed, its destructor is called first, followed by the destructors of its base classes. This ensures that any resources allocated by derived classes are properly released before the base class destructors are invoked

True or False?

1. When using public inheritance, the public members of the base class become private in the derived class.
2. A constructor must always initialize all member variables of a class in C++.
3. A destructor in C++ can be explicitly called using the class name and the destructor name.
4. In C++, member variables of a class can be initialized directly within the class definition.
5. A class can have multiple destructors if they have different parameter lists.

True or False?

6. A recursive function in C++ can call itself directly or indirectly through other functions.

7. Recursive functions must always make progress towards the base case in each recursive call.

Explain...

- What can cause a memory leak?
- What is the purpose of encapsulation and how do we use it?
- How do we initialize/access private data members?