CPSC 256 – Midterm Test 4

The test consists of two questions, each worth 15 points. You should complete your solutions on the test paper itself. You are allowed pens, pencils, erasers, and blank scrap paper. No other aids are allowed.

Question 1: Inheritance and Polymorphism

You are building a program to manage different types of shapes. The base class is Shape, and each derived class represents a specific shape (e.g., Circle and Rectangle). The Shape class contains a virtual method getArea() that is implemented differently in each derived class.

1. (5 points)

Write the declaration of the Shape base class, including:

• A protected member variable for the shape’s name.

• A public virtual method double getArea() = 0; to make the class abstract.

• A non-virtual method void printName() to print the shape’s name.

• An initialization constructor which sets the internal name variable to the

provided shape name, and a default destructor.

1. (10 points)

Implement two derived classes:

• Circle: It should have a radius member variable and

override the getArea() method to calculate the area of the circle.

• Rectangle: It should have width and height member

variables and override the getArea() method to calculate

the area of the rectangle.

The main() function is provided below:

int main() {

// Create objects of derived classes

Shape\* shapes[2]; // Array of pointers to Shape

shapes[0] = new Circle(5.0); // Circle with radius 5

shapes[1] = new Rectangle(4.0, 6.0); // Rectangle with width 4 and height 6

// Demonstrate polymorphism

for (int i = 0; i < 2; i++) {

shapes[i]->printName(); // Calls the non-virtual method

cout << "Area: " << shapes[i]->getArea() << endl; // Calls the overridden method

}

// Free dynamically allocated memory

for (int i = 0; i < 2; i++) {

delete shapes[i];

}

return 0;

}

#include <iostream>

#include <cmath> // For M\_PI in Circle area calculation

using namespace std;

// Abstract base class

class Shape {

};

// Derived class: Circle

class Circle : public Shape {

};

// Derived class: Rectangle

class Rectangle : public Shape {

};

Question 2: Linked Lists

I asked ChatGPT to generate a question that would test your knowledge of linked lists in C++. Here is the question that it came up with , together with the solution that it generated:

You are tasked with designing a simple singly linked list to store integers. Write a program that creates a linked list and appends the numbers 10, 20, and 30 to the list, and then displays the list.

#include <iostream>

using namespace std;

struct Node {

int data; // TODO1: define internal Node variable to hold the integer data

Node\* next; // TODO2: define internal pointer to the next node in the linked list

// TODO3: Initialization “constructor” for Node objects

Node(int value) : data(value), next(nullptr) {}

};

class LinkedList {

private:

Node\* head; // TODO4: define a pointer to the head node of the linked list

public:

// TODO5: Default constructor

LinkedList() : head(nullptr) {}

// TODO6: add a new node with datavalue = value to the end of the linked list

void append(int value) {

Node\* newNode = new Node(value);

if (head == nullptr) { // TODO7: check to see if there are any nodes in the list already

head = newNode;

} else {

Node\* temp = head;

while (temp->next != nullptr) { // TODO8: search through this list for nullptr – this tells us we are at the end

temp = temp->next;

}

temp->next = newNode; // TODO9: Set the next pointer of the original last node equal to the address of new node

}

}

// TODO:10. : Print out the entire linked list

void display() {

Node\* temp = head;

while (temp != nullptr) { // TODO11: keep printing the nodes until we get to the end (next = nullptr)

cout << temp->data << " -> ";

temp = temp->next;

}

cout << "NULL" << endl; // TODO12: After printing all the nodes, print “NULL”

}

// TODO13: LinkedList Destructor

~LinkedList() {

Node\* temp;

while (head != nullptr) {

temp = head;

head = head->next;

delete temp;

}

}

};

int main() {

LinkedList list;

// TODO14: Create three nodes in the linked list with values 10, 20, and 30

list.append(10);

list.append(20);

list.append(30);

// TODO15: Print the entire linked list

cout << "The linked list contains: ";

list.display();

return 0;

}

Your task is to complete the comments in the above code (indicated in 15 places with TODO) so as to demonstrate that you understand how the code works.