CS2810 OOAIA

Classes and Objects

Example Code Snippets

1. **Basic Class and Object**

```cpp
#include <iostream>

using namespace std;

// Defining a class
class Car {
public:
    string model;
    int year;
    void display() {
        cout << "Model: " << model << ", Year: " << year
<< endl;
    }
};

int main() {
    Car car1;  // Creating an object
    car1.model = "TATA";
    car1.year = 2025;
    car1.display();  // Calling a function
    return 0;
}
```

2. **Encapsulation with Private Data Members**

```
#include <iostream>

using namespace std;

class Student {
private:
    string name;
    int age;
public:
    void setData(string n, int a) {
        name = n;
        age = a;
    }
    void display() {
        cout << "Name: " << name << ", Age: " << age <<
endl;
    }
};

int main() {
    Student s1;
    s1.setData("Alice", 20);
    s1.display();
    return 0;
}
```

3. **Constructor and Destructor**

```
#include <iostream>

using namespace std;

class Employee {
public:
    Employee() { // Constructor
```

```
        cout << "Employee object created" << endl;
    }

    ~Employee() { // Destructor
        cout << "Employee object destroyed" << endl;
    }
};

int main() {
    Employee e1;  // Constructor is called automatically
    return 0;  // Destructor is called when object goes out
of scope
}
```

4. **Access Specifiers: Public, Private, and Protected**

```
#include <iostream>

using namespace std;

class Demo {
public:
    int publicVar;
protected:
    int protectedVar;
private:
    int privateVar;
};

int main() {
    Demo d;
    d.publicVar = 10;  // Accessible
    // d.protectedVar = 20;  // Error: Not accessible
    // d.privateVar = 30;    // Error: Not accessible
    return 0;
}
```

5. **Static Members in a Class**

```cpp
#include <iostream>

using namespace std;

class Counter {
private:
    static int count;  // Static variable
public:
    Counter() { count++; }
    static void showCount() {  // Static function
        cout << "Count: " << count << endl;
    }
};

int Counter::count = 0; // Initialize static variable

int main() {
    Counter c1, c2, c3;
    Counter::showCount();  // Call static function
    return 0;
}
```

6. **Class with Parameterized Constructor**

```cpp
#include <iostream>

using namespace std;

class Rectangle {
private:
    int length, width;
public:
    Rectangle(int l, int w) {  // Parameterized
constructor
```

```
        length = l;
        width = w;
    }
    int area() {
        return length * width;
    }
};

int main() {
    Rectangle r1(5, 3);  // Passing values during object
creation
    cout << "Area: " << r1.area() << endl;
    return 0;
}
```

Practice Problems

1. **Time Class Implementation**
    - Create a Time class with attributes: hours, minutes, and seconds.
    - Implement a method to add two Time objects.
    - Display the time in a 24-hour format.

2. **Matrix Class with Operations**
    - Design a Matrix class that supports basic matrix operations:
    - Addition of two matrices.
    - Multiplication of two matrices.
    - Implement proper constructors and destructors for memory management.

3. **Smart Pointer Implementation**
    - Implement a SmartPointer class that manages dynamic memory allocation.

- Ensure memory is automatically deallocated when the pointer goes out of scope.
- Functionality should be similar to unique_ptr in C++.

4. **Student Records Management**
    - Create a Student class with attributes like name, roll number, and marks.
    - Implement a method to display student details.
    - Use a static variable to maintain count.
    - Maintain an array of students and provide a function to find the student with the highest marks.