Course Name:	Object Oriented Programming	Course Code:	CS1004
Degree Program:	BS (CS, SE, DS, Robotics)	Semester:	Fall 2023
Exam Duration:	180 Minutes	Total Marks:	70
Paper Date:	26-12-2023	Weight	40
Section:	ALL	Page(s):	10
Exam Type:	Final Exam		

Student Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes: Attempt all questions. Answer in the space provided. **Answers written on rough sheet will not be attached and marked**. Do not use pencil or red ink to answer the questions**.**

Question 1: (CLO: 1) **(Marks: 10)**

Determine output for the code segment given below. There is no syntax error in the code.

```
class FirstClass {
public:
    int val1;
    FirstClass(int value=0) : val1(value) {
        cout << "FirstClass Constructor with value: " << value << endl;
    }
    void display() {
        cout << "Displaying from FirstClass: " << val1 << endl;    }
    ~FirstClass() {
        cout << "Destructor called for FirstClass " << val1 << endl;    }
};
class SecondClass {
    FirstClass object;
public:
    int val2;
    SecondClass(int value) : val2(value), object(value + 5) {
        cout << "SecondClass Constructor with value: " << val2 << endl;
    }
    SecondClass(SecondClass& obj)
    {
        cout << "SecondClass Copy Constructor" << endl;
        this->val2 = obj.val2+10;
    }
    void display() {
        object.display();
        cout << "Displaying from SecondClass: " << val2 << endl;    }
    ~SecondClass() {
        cout << "Destructor called for SecondClass " << val2 << endl;    }
};
void functionDisplay(SecondClass &s, FirstClass f) {
    s.display();
    f.display();
}
int main() {
    SecondClass secondObj(10);FirstClass firstObj(20);
    functionDisplay(secondObj, firstObj);
}
```

OUTPUT:

Question 2: (CLO: 4) **(Marks: 5+5)**

Determine output for the codes given below. There is no syntax error in the code.

a)

```
int main() {
    try {
        try {
            throw 20;
            cout << "Hello from try block\n";
        }
        catch (int n) {
            cout << "Exception catched\n";
            throw;
        }
    }
    catch (float n) {
        cout << n << ": float Exception catched\n";
    }
    catch (int n) {
        try {
            cout << n << ": int Exception catched\n";
            throw exception("Exp in catch block");
        }
        catch (exception n) {
            cout << n.what() << "\n";
        }
    }
    catch (...) {
        cout << "Exception catched\n";
    }
    cout << "Bye!";
    return 0;
}
```

OUTPUT:

b)

```
template <typename TYPE>
class Carray
{
private:
    TYPE x, y;
public:
    Carray(const TYPE a, const TYPE b) : x(a), y(b) {}
    TYPE getX() { return x; }
    TYPE getY() { return y; }
};
template <class TypeName>
void fun(TypeName* arr, int len)
{
    for (int i = 0; i < len; i++) {
        cout << *arr << ", ";
        ++arr;
    }
    cout << endl;
}
int main()
{
    Carray<float>f(200.5, 34);
    cout << f.getX() <<" " << f.getY() << endl;
    char chrArr[] = "Help";
    int numArr[] = { 10, 20, 30, 40, 50 };
    cout << "chrArr: ";
    fun(chrArr, strlen(chrArr));
    cout << "numArr: ";
    fun(numArr, 5);
    cout << "Hello: ";
    fun("Hello", 5);
    system("pause");
    return 0;
}
```

OUTPUT:

Question 3: (CLO: 3) **(Marks: 10)**

You are designing a library management system that needs to efficiently allocate books based on their genres and availability. The library receives a static 2D array representing the number of books available for different genres in each section. Your task is to create a dynamic 2D array that organizes these books according to the following conditions:

- Each row in the static array represents a section of the library.
- Each column in a row represents a genre of books available in that section.
- The value at **staticArray[section][genre]** denotes the number of books available for that particular genre in that section.

Your program needs to perform the following tasks:

- Identify the sections where the total number of books available across all genres is greater than a given threshold (**thresholdBooks**).
- Create a Dynamic array for these identified sections, where each row corresponds to a section having more books than the threshold.
- Allocate memory dynamically for these sections and genres and populate the Dynamic array accordingly.
- Ensure that memory is deallocated appropriately after use.

Write a C++ program that takes the static 2D array as input, applies the conditions mentioned above, and outputs the 2D Dynamic array.

Here are the additional details:

- Use double pointers (**) for creating the dynamic array.
- Implement functions to handle memory allocation, population of the Dynamic array, and deallocation.

Your program should contain the following functions:

// Function to create the dynamic array based on conditions

int createDynamicArray(int staticArray[][3], int sections, int genres, int thresholdBooks, int& DynamicArrayRows);**

// Function to populate the dynamic array with books based on conditions

void populateDynamicArray (int staticArray[][3], int DynamicArray, int sections, int genres, int thresholdBooks);**

// Function to deallocate memory used by the dynamic array

void deallocateDynamicArray(int DynamicArray, int DynamicArrayRows);**

Example Input:

```
int staticArray[4][3] = {
   {10, 5, 7},
   {3, 12, 8},
   {2, 1, 9},
   {2, 5, 3}};
int sections = 4;
int genres = 3;
int thresholdBooks = 15;
```

Example Output:

Dynamic Array for Sections with More than 15 Books:
Section 1: 10 5 7
Section 2: 3 12 8

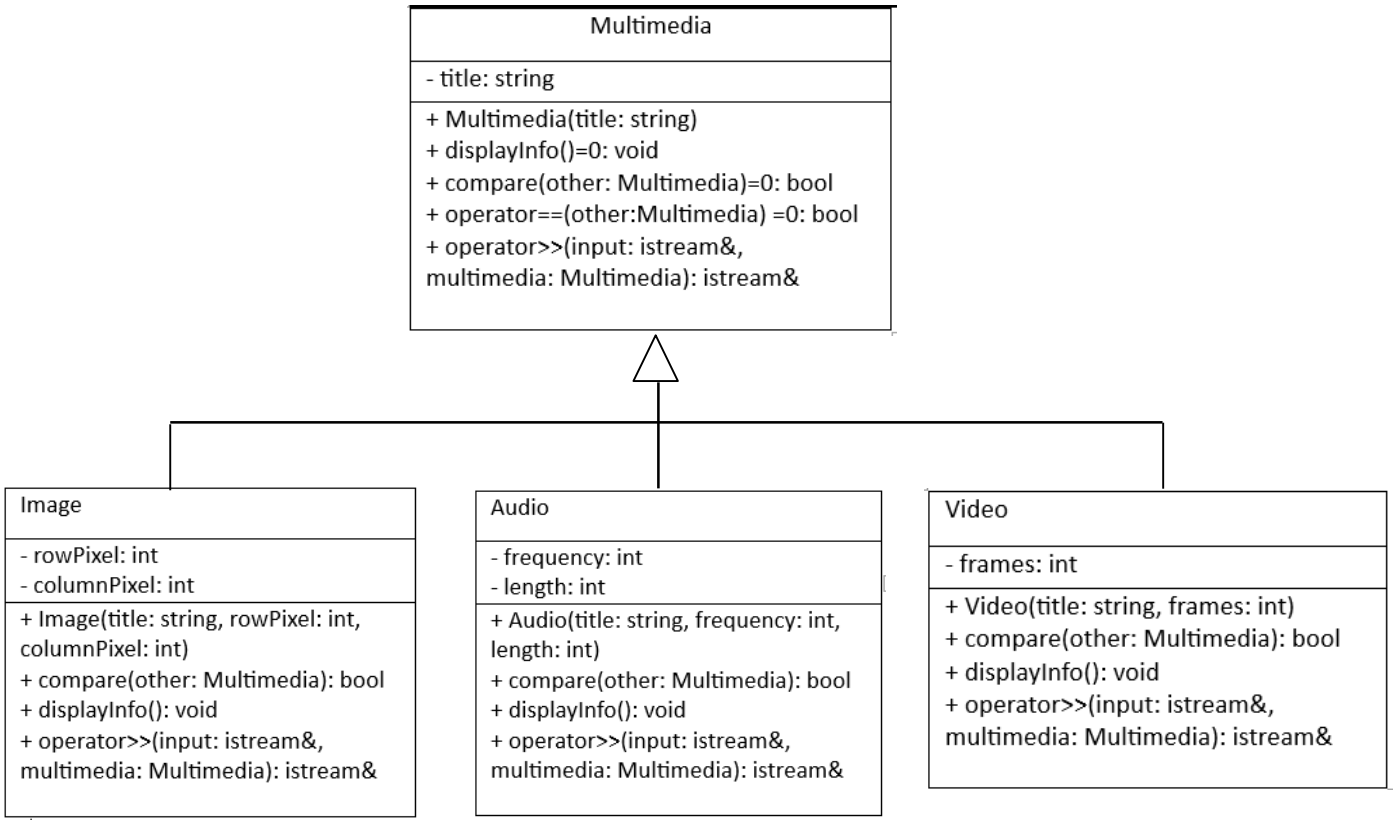
Main Function is given, your code should work for this main,

```
int main() {
    int staticArray[4][3] = {
        {10, 5, 7},
        {3, 1, 8},
        {6, 4, 9},
        {2, 5, 3}
    };
    int dynamicArrayRows = 0;
    int sections = 4;
    int genres = 3;
    int thresholdBooks = 15;
    // Creating the dynamic array
    int** dynamicArray = createDynamicArray(staticArray, sections, genres, thresholdBooks,
dynamicArrayRows);
    // Populating the dynamic array
    populateDynamicArray(staticArray, dynamicArray, sections, genres, thresholdBooks);
    // Displaying the result
    cout << "Dynamic Array for Sections with More than " << thresholdBooks << " Books:\n";
    for (int i = 0; i < dynamicArrayRows; ++i) {
        for (int j = 0; j < genres; ++j) {
                cout << dynamicArray[i][j] << " ";
            }
            cout <<endl;
        }
    // Deallocating memory
    deallocateDynamicArray(dynamicArray, dynamicArrayRows);
    return 0;
}
```

Question 4: (CLO: 2) **(Marks: 15)**

Implement the following UML diagram using concept of abstract class and polymorphism without changing main function.

Note: you can typecast base class object to derived class object in compare function
e.g; **Image* imageObj = (Image*)other;**




```
You Code should run for this main().
int main() {
    Image image("Default Image", 800, 600);
    Audio audio("Default Audio", 44100, 180);
    Video video("Default Video", 24);
    cin >> image;
    cin >> audio;
    cin >> video;
    Multimedia* multimediaArray[] = { &image, &audio, &video };
    for (int i = 0; i < 3; i++) {
        multimediaArray[i]->displayInfo();
        if (multimediaArray[i]->compare(image)) {
            cout << "The content is similar to the image." << endl;
        }
        else {
            cout << "The content is different from the image." <<endl;
        }

        if (*multimediaArray[i] == audio) {
            cout << "The content is equal to the audio." << endl;
        }
        else {
            cout << "The content is not equal to the audio." << endl;
        }
        cout << "------------------------" << endl;
    }
    return 0;
}
```

Question 5: (CLO 1) Short Questions: **(Marks 5+5+5+5+5=25)**

1. Read the following code, A double pointer array is declared named person.

Person persons = new Person*[5];**

Complete the code to resize the array to add 7 people to this array

```
class Person {
public:
    string name;
    int age;
    Person(string n, int a) : name(n), age(a) {}
};
int main() {
    int capacity = 5;
    // Creating an array of 5 pointers to Person objects
    Person** persons = new Person * [capacity];
    // Adding 5 people
    for (int i = 0; i < capacity; ++i) {
        persons[i] = new Person("Person", 25 + i);
//Write your code for resizing array here, your code should run perfect with the already
written lines of code:









    // Displaying the added people
    for (int i = 0; i < capacity; ++i) {
        cout << persons[i]->age <<endl;
    }
    for (int i = 0; i < capacity; ++i) {
        delete persons[i];
    }
    delete[] persons;
}
```

2. What is the difference between concrete and final class? And what is the purpose of using final keyword with the function name in class?

3.Write output of the following code:

```
class ClassA{
private:
    int x, y;
public:
    static int w;
    ClassA(int x = 0, int y = 0) {
        this->x = x;
        this->y = y; w - 1;
    }
    ClassA* setX(int a) {
        x = a; w++;
        return this;
    }
    ClassA* setY(int b) {
        y = b;
        return this;
    }
    void print() {
        cout << "x" << x << "y" << y << endl;
    }
    static void func(ClassA b) {
        cout <<w << endl;
    }
};
int ClassA::w = 1;
```

```
int main(){
    ClassA* obj1 = new ClassA(5, 5);
    (*(*obj1).setX(30)).setY(50);
    obj1->print();
    cout << ClassA::w << endl;
    ClassA::func(*obj1);
    return 0;
}
```

Output:

4.Find the error in this code and explain the reason:

```
class Base {
public:
    Base(string& baseData) :
baseData(baseData) {}
    virtual void displayInfo() const {
        cout << "Base Class: " << baseData;
    }
private:
    string baseData;
};
class Derived : public Base {
public:
    Derived(string baseData, string
derivedData)
: Base(baseData), derivedData(derivedData) {}
    virtual void display() const {
        cout << "Derived Class: " <<
derivedData;
    }
private:
    string derivedData;
};
int main() {
    Derived derivedObject("Base Data",
"Derived Data");
    Base  base = derivedObject;
    base.display();
}
```

Answer:

5.Write output of the following code:

```
class Base {
public:
    Base() {
       cout << "Base Constr" << endl;
    }
    ~Base() {
        cout << "Base Destr" << endl;
    }
};
class Derived1 : public Base {
public:
    Derived1() {
        cout << "Derived1 Constr" << endl;
    }
    ~Derived1() {
        cout << "Derived1 Destr" << endl;
    }
};
int main() {
    Base* Basepointer = new Derived1;
    {
        Derived1 obj;
        delete Basepointer;
    }
}
```

Output: