GURU TEGH BAHADUR INSTITUTE OF TECHNOLOGY

SUBJECT NAME : OBJECT ORIENTED PROGRAMMING LAB

SUBJECT CODE : CIC-211

SEMESTER :- THIRD

		L	T	P	C
BCS3L2	**OBJECT ORIENTED PROGRAMMING LAB**	**L**	**T**	**P**	**C**
	Total Contact Hours - 30	0	0	3	2
	Prerequisite –Fundamental of Computing and Programming, Object Oriented Programming using C++,C.				
	Lab Manual Prepared by – Dept. of Computer Science & Engineering				

OBJECTIVES: This lab manual demonstrates familiarity with various concepts of OOPS.

COURSE OUTCOMES (COs)

CO1	Demonstrate class object concepts by using C++.
CO2	Develop programs using inheritance and polymorphism.
CO3	Demonstrate the significance of constructors and destructor.
CO4	Implement function and operator overloading using C++.
CO5	Construct generic classes using template concepts.
CO6	Implement the concept of file handling.

MAPPING BETWEEN COURSE OUTCOMES & PROGRAM OUTCOMES
(3/2/1 INDICATES STRENGTH OF CORRELATION) 3- High, 2- Medium, 1-Low

COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	2	3	3		3	2	2	2	3	3	3	3		3	
CO2									2					3	
CO3		3	3	2	2		3		3		3	3		3	
CO4	2					1		2		2				3	
CO5			2		3		2		2		2	2	2	3	
CO6	1	2	2		2	2		1	2	2	3			3	
Category	Professional Core (PC)														
Approval	37th Meeting of Academic Council, May 2015														

LIST OF EXPERIMENTS:

1. Programs Using Functions
- Functions with default arguments
- Implementation of Call by Value, Call by Address and Call by Reference
2. Simple Classes for understanding objects, member functions and Constructors
- Classes with primitive data members
- Classes with arrays as data members
- Classes with pointers as data members – String Class
- Classes with constant data members, Classes with static member functions
3. Compile time Polymorphism
- Operator Overloading including Unary and Binary Operators, Function Overloading
4. Runtime Polymorphism
- Inheritance ,Virtual functions
- Virtual Base Classes, Templates
- File Handling-Sequential access, Random access.

GURU TEGH BAHADUR INSTITUTE OF TECHNOLOGY

<u>OBJECT ORIENTED PROGRAMMING LAB (CIC -211)</u>

LIST OF PRACTICALS

<u>CLASSES AND OBJECTS</u>

1. A class student has three data members: name, roll, marks of 5 subjects and member functions to assign streams on the basis of the table given below:

Average marks	**Stream**
96% and more	computer science
91% - 95%	electronics
86% - 90%	mechanical
81% - 85%	electrical
76% - 80%	chemical
71% - 75%	civil

Declare the class student and define the member functions.

2. Declare a class to represent bank account of 10 customers with the following data members:

Name of depositor

Account number

Type of account (s for savings, c for current)

Balance amount

The class also contains the following member functions:

A. To initialize

B. To deposit money

C. For withdrawal (if the deposit after withdrawal is greater than 10000)

D. To display the data members

3. Define a class employee with the following specifications:

Private members of class employee:

Empno

Ename

Basic

Hra = 10% of basic

Da = 20% of basic

Netpay

Calculate()

Public members of class employee:

Havedata()

Dispdata()

4. Develop a Program to enter traveling details and tell number of buses required using classes and objects.

5. Demonstrate use of scope resolution operator using multiple initializations of the variable.

6. Write a program for multiplication of two matrices using OOP.

7. Use inline functions and macros to obtain the largest of three numbers.

8. Register the entrance of people in the auditorium using static class data

9. Write a program to find the greatest of two given numbers in two different classes using friend function.

10. Create a class called Date, with integer data members for day, month and year. The class comprises of member functions

(1) To display date in DD/MM/YYYY format.

(2) To subtract an integer from date object

(3) To subtract one date from another.

11. A hospital wants to create a database regarding its indoor patients. The information to store includes

(a) Name of patient

(b)Date of admission

(c)Disease

(d)Date of discharge

Use the Date class created in previous program to store the date. The patient class comprises of The member functions to enter the information and display the list of all patients in database

CONSTRUCTORS AND DESTRUCTORS

12. Define a class Serial with following specifications:

Private members:

serialCode	integer
title	20 characters
duration	float
noOfEpisodes	integer

Public member function of class Serial:

1. A constructor to initialize duration as 30 and noOfEpisodes as 10

2. newSerial () to accept values of serial code and title

3. otherEntries () to assign value to duration and noOfEpisodes with the help of values passed to the function

4. dispData () to display the data members on the screen

13. Considering the following specifications:

Structure name

Name

First	char [40]
Mid	char [40]
Last	char [60]

Structure name

Phone

Area	char [4]
Exch	char [4]
Numb	char[6]

Class name, p_rec with data mambers as objects of structure name and phone with member functions and constructor.

14.Define a class student with the following specifications:

Private members:

Roll_no

Name

Class_st

Marks

Percentage

Calculate()

Public members:

Readmarks() which reads the marks and invokes the calculate function

Displaydata() which prints the data.

15. Declare a class String. It must have constructors which allow definition of object in the following form (the class string has data members str of type char *):

String name1;//str point to NULL

String name2=”ABC”;//one argument constructor is invoked

String name3=name2;//one argument constructor taking string object

Writ a program to model string class and to manipulate its objects.The destructor must release memory allocated to str data members by its counter part.

16. Write a program to perform addition of two complex numbers using constructor overloading. The first

constructor which takes no argument is used to create objects which are not initialized, second which

takes one argument is used to initialize real and imag parts to equal values and third which takes two

argument is used to initialized real and imag to two different values.

INHERITANCE

17. Create 2 classes namely student and exam. Make the derived class result to inherit the details of total- marks and students through multilevel inheritance.

18. Implement the above program using multiple inheritance.

19. Define a class to store coordinates of a point with member function to read the coordinates and display the coordinates. Define a derived class with the additional capability to store the distance of the point from the origin. Write the additional member functions for the same. Write a program, using the classes defined above to read coordinates of a point and find its distance from the origin.

20. Imagine a publishing company that markets both book and audio cassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication from this class derived two classes : book, which adds a page count (type int) ; and tape , which adds a playing time in minutes (type float). Each of these classes should have getdata() and a putdata(). Write a main program to test the book and tape classes by creating instances of them and asking a user to fill in their data with getdata () and displaying the data with putdata().

21. Use Patient class created above and create a derived class to store the age of patients. Display the list of all pediatric patients less than 12 yrs. of age by using member function in the derived class.

COMPILE TIME AND RUN TIME POLYMORPHISM

22. Design a program for calculating the area of a triangle, rectangle and circle by taking shape as the base class using virtual functions.

23. Create a class called List with two pure virtual function store() and retrieve().To store a value call store and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and retrieve.

24. Write a program to overload function area() to find area of triangle using heroes formula, area of rectangle , area of square and area of circle.

OPERATOR OVERLOADING

25. Write a program overloading unary operator to increment date.

26. Write a program overloading arithmetic operators to add two complex numbers.

27. Implement a class string containing the following functions:

- Overload + operator to carry out the concatenation of strings.
- Overload = operator to carry out string copy.
- Overload <= operator to carry out the comparison of strings.
- Function to display the length of a string.
- Function tolower( ) to convert upper case letters to lower case.
- Function toupper( ) to convert lower case letters to upper case.

28. Write a program overloading new and delete operator.

FILE HANDLING

29. Program to read and write data in a text file using fstream only.

30. Program to display word by word data from file.

31. Program to count total number of words and spaces in a file.

32. Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space ,line feed ,new line and carriage return from a text file and store the contents of the file without the white spaces on another file.

33. Write a program to read the class object of student info such as name , age ,sex ,height and weight from the keyboard and to store them on a specified file using read() and write() functions. Again the same file is opened for reading and displaying the contents of the file on the screen.

34. Program to enter data into Hotel file using class, and count the total number of customers.

35. Program on merging of records from 2 files.

TEMPLATES

36. Write a program for creating doubly linked list. The doubly linked list class must be of template type.

37. Write a program to define the function template for calculating the square of given numbers with different data types.

38. Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.

EXCEPTION HANDLING

39. Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond the array size.

40. Write a program to develop a User Defined Exception.

Dev C++

Editing a file

Editing a file is pretty straightforward. Create (Ctrl+N) or open (Ctrl+O) a file.

Tips

- Hover over a variable or include line to show information about it.
- Click on a line number to place a breakpoint there.
- Triple click on a line to select it.
- Code completion can be opened using Ctrl+Space anywhere. It will show valid statements found in the current file, its included files and the cache. Preferences can be set at Tools >> Editor Options >> Completion >> Code Completion.
- Symbol completion can be customized at Tools >> Editor Options >> Completion >> Symbol Completion.
- Consider caching frequently included files to speed up file and project opening.

Compiling a source file

A special feature of Dev-C++ is the ability to compile (and run) single source files. This is done via the following steps:

- Create (Ctrl+N) or open (Ctrl+O) a source file.
- Click Compile (F9) to compile the currently visible file.
- *Optionally, run the created executable (F10).*

This compilation will use the compiler set selected at Tools >> Compiler Options.

Compiling a project

Compiling a project largely requires the same steps:

- Create (Ctrl+N) or open (Ctrl+O) a project.
- Click Compile (F9) to apply changes made in a source file.

- Click Rebuild (F12) to apply changes made in a header file or to recompile all files.
- *Optionally, run the created executable (F10).*

This compilation will use the compiler set selected at Tools >> Compiler Options, but unlike source file compilations, it will use the compiler **settings** located at Project >> Project Options >> Compiler.

Tips

- For single file compilations, make sure to use proper file extensions. When not explicitly telling the compiler which standard to use (using the -std flag), use *.c to compile as C or *.cpp to compile as C++.
- To reduce compilation output size, especially when using C++ headers, consider passing -s to the compiler. This can be done manually or via (Settings) >> Linker >> Strip executable. This option will remove any unused data from the output. A downside of this option is that debugging and profiling will not work anymore. An alternative but much less effective approach is to reduce the amount of included files.
- To reduce compilation time, consider including less files. For example, when using a project, don't blindly include every file in every file!
- To save one click in the compilation process, try out Compile & Run (F11). This will, obviously, compile and run your file or project immediately afterwards.
- When changing a header file, make sure to recompile all source files that make use of it by rebuilding. Not doing so will lead to undefined executable behaviour: the code you're seeing will not be what will be executed because some source files are making use of outdated headers.

Debugging a source file

Debugging a source file can be done using the following steps:

- Create (Ctrl+N) or open (Ctrl+O) a source file.
- At Tools >> Compiler Options, select a compiler set that supports debugging (-g3).
- Click Debug (F5) to compile and debug the currently visible file.
- Place a breakpoint by clicking on a line number or use F4 to pause your program and inspect it.

Debugging a project

Debugging a project file can be done using the following steps:

- Create (Ctrl+N) or open (Ctrl+O) a project.
- At Project >> Project Options >> Compiler, select a compiler set that supports debugging (-g3).
- Click Debug (F5) to compile and debug the project.
- Place a breakpoint by clicking on a line number or use F4 to pause your program and inspect it

Tips

- When changing compiler sets, make sure to rebuild before debugging. Dev-C++ will only suggest doing so if a source file is newer the executable, but not when the compiler changes!
- Optimization (-Ox) will, among other effects, change the order of execution and can remove code that yields no net change. This can be quite annoying when walking through code line by line. Consider debugging without optimization or use -Og to fix this.

  http://gcc.gnu.org/onlinedocs/gnat_ugn_unw/Debugging-Optimized-Code.html

- Adding debug information usually more than doubles compilation time. Consider disabling it to lower compilation times.
- The strip flag (-s) will remove debugging information added by -g3. Consider disabling it.
- Debugging information adds considerable size (an order of magnitude isn't out of the ordinary) to created programs. Consider disabling it to reduce file size.

Profiling a source file

Profiling a source file can be done using the following steps:

- Create (Ctrl+N) or open (Ctrl+O) a source file.
- At Tools >> Compiler Options, select a compiler set that supports profiling (-pg).
- Click Profile Analysis (next to the Debug button) to compile and profile the currently visible file.
- When closing the created program, a timing analysis will open.

Profiling a project

Profiling a project file can be done using the following steps:

- Create (Ctrl+N) or open (Ctrl+O) a source file.
- At Tools >> Compiler Options, select a compiler set that supports profiling (-pg).
- Click Profile Analysis (next to the Debug button) to compile and profile the project.
- When closing the created program, a timing analysis will open.

Tips

- **Only functions that consume more than 0.01 second of CPU time will show up in the analysis.**
- When changing compiler sets, make sure to rebuild before profiling.
- Optimization (-Ox) can remove code that yields no net change. One should be careful when profiling these kinds of code:

  ```
  tbefore = GetTime();
  DoSomething();
  tdiff = GetTime() - tbefore;
  ```

  If the function DoSomething() does not affect output in any way and when other code is independent from it, the call to it can be removed by the compiler and tdiff will be zero! Consider printing something calculated inside DoSomething() after calculating tdiff. This way, other code will be dependent on DoSomething() because you print some result of it afterwards.

- The strip flag (-s) will remove profiling information added by -pg. Consider disabling it.

Frequenty Asked Questions

Q: What compilers does Dev-C++ support?
A: Currently, any port of GCC to Windows is supported. This includes MinGW, MinGW-w64 and TDM-GCC. Clang should work too, but you'll have to port that to Windows yourself or find someone else who did it for you.

Q: How do I add a compiler to Dev-C++?
A: All compiler settings are managed at Tools >> Compiler Options. One can add a set in the following ways:

- Use "Add compiler set by folder" and point Dev-C++ to the folder where your compiler is installed. Dev-C++ will then attempt to configure it by asking the following program about the compiler folder layout:

  ```
  \bin\gcc.exe
  ```

- Use "Add an empty set" and fill out the options yourself.

Q: I want to compile for 32-bit with my TDM-GCC x64 install. How do I do that?
A: The simplest way to do that is to select a preconfigured compiler set that will compile for 32bit. Go to Tools >> Compiler Options and check if the list contains any. If it does, select it and click OK. Repeat this step for each project at Project >> Project Options >> Compiler.
To manually create such a compiler set, you need to do two things:

- Add '-m32' to the compiler command. Use "Compiler with the following pointer width" at "Settings >> Code Generation" or type it in a custom command at "General >> Add the following commands when calling the compiler".
- Use different libraries. Go to Directories >> Libraries and replace the "(..)lib" directories with "(..)lib32".

Q: I'm specifically compiling for 32bit or 64bit, but GCC is throwing errors at me!
A: If it looks like this:

```
skipping incompatible (directory)/libmingw32.a when searching for -lmingw32
skipping incompatible (directory)\libmingw32.a when searching for -lmingw32
skipping incompatible (directory)/libmoldname.a when searching for -lmoldname
skipping incompatible (directory)\libmoldname.a when searching for -lmoldname
...
```

... then you need to change the library directory. Go to Tools >> Compiler Options >> (select the current compiler) >> Directories >> Libraries. There, use "(...)lib" folders for 64bit, or (...)lib32 for 32bit".
Q: Can I disable these annoying mouseover popups?
A: Sure. You can toggle them in Tools >> Editor Options >> Show Editor Hints.

Q: I'm using the TDM-GCC x64 version and cache creation is taking longer than forever. What can I do?
A: Yes, it's taking very long, thanks to the huge amount of headers supplied by TDM-GCC x64. Of course, no single programmer will ever use them all. I suggest selecting the custom caching options or navigating to Tools >> Editor Options >> Class Browsing >> Completion. Add the headers **you** often use. You don't have to add everything manually fortunately. Adding windows.h for example will include the whole massive heap of Microsoft stuff referenced inside that file.

Q: Programs created with Dev-C++ show the line "Process exited with return value X". What's going on?
This piece of text is shown when a console program ran via Dev-C++ exits. The host that runs your console program (ConsolePauser.exe) prints this code. **Dev-C++ does NOT add any code to your programs to print this.** This feature is quite useful because you can see console output before closing. You can remove it by unticking "Tools >> Environment Options >> General >> Pause console programs after return".

SAMPLE PROGRAMS

Program 1: Hello, World!

```
#include <iostream>



int main() {

   std::cout << "Hello, World!" << std::endl;

   return 0;

}
```

Question 1: What is the purpose of the #include <iostream> statement in the program?

Answer 1: The #include <iostream> statement includes the necessary header file to enable input and output stream functionality in C++.

Program 2: Simple Calculator

```
#include <iostream>



int main() {

   double num1, num2;

   char op;



   std::cout << "Enter two numbers: ";

   std::cin >> num1 >> num2;
```

```
std::cout << "Enter an operator (+, -, *, /): ";
std::cin >> op;

double result;
switch (op) {
    case '+':
        result = num1 + num2;
        break;
    case '-':
        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        result = num1 / num2;
        break;
    default:
        std::cout << "Invalid operator";
        return 1;
}

std::cout << "Result: " << result << std::endl;
return 0;
```

```
}
```

Question 2: What does the switch statement do in the program, and how is it used?

Answer 2: The switch statement is used to evaluate the value of the op variable (operator) and perform corresponding arithmetic operations on num1 and num2. It allows the program to choose the appropriate case based on the user's operator input.

Program 3: Factorial Calculation using Recursion

```
#include <iostream>


int factorial(int n) {

  if (n == 0 || n == 1)

      return 1;

  else

      return n * factorial(n - 1);

}


int main() {

  int num;

  std::cout << "Enter a number: ";

  std::cin >> num;


  std::cout << "Factorial of " << num << " is: " << factorial(num) << std::endl;

  return 0;
}
```

Question 3: Explain how the factorial function works and why recursion is used in this program.

Answer 3: The factorial function calculates the factorial of a given number n. Recursion is used to break down the problem into smaller subproblems by repeatedly calling the factorial function with a smaller value of n. The base case is when n is 0 or 1, where the function returns 1. By using recursion, the function accumulates the product of n and the factorial of n-1, effectively calculating the factorial of the given number.

Program 4: Class and Object

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

class Student {
public:
    std::string name;
    int rollNumber;

    void display() {
        std::cout << "Name: " << name << std::endl;
        std::cout << "Roll Number: " << rollNumber << std::endl;
    }
};

int main() {
    Student s;
    s.name = "John";

    s.rollNumber = 101;



    std::cout << "Student Information:" << std::endl;

    s.display();



    return 0;

}
```

Question 4: Define a class named Student with attributes name and rollNumber. How is an object of this class created and its attributes accessed?

Answer 4: The class Student is defined with two attributes: name and rollNumber. An object s of the class is created using Student s;, and its attributes are accessed and modified using the dot (.) operator, such as s.name and s.rollNumber.

Program 5: Constructor and Destructor

```
#include <iostream>

#include <string>



class Book {

public:

    std::string title;

    std::string author;



    Book(std::string t, std::string a) {
        title = t;

        author = a;

        std::cout << "Constructor called for " << title << std::endl;

    }



    ~Book() {

        std::cout << "Destructor called for " << title << std::endl;

    }

};



int main() {

    Book book1("The Great Gatsby", "F. Scott Fitzgerald");

    Book book2("To Kill a Mockingbird", "Harper Lee");



    return 0;

}
```

Question 5: Explain the purpose of the constructor and destructor in the Book class. How are they called in the program?

Answer 5: The constructor is used to initialize the attributes of an object when it is created. In this case, the constructor takes title and author as parameters and initializes the corresponding attributes. The destructor is called when an object goes out of scope or is explicitly destroyed. In the program, the constructor is called when book1 and book2 are created, and the destructor is called when the program exits.

Program 6: Inheritance and Function Overriding

```
#include <iostream>
```

```
#include <string>


class Shape {

public:

    virtual double area() {

        return 0;

    }

};


class Circle : public Shape {

private:

    double radius;


public:

    Circle(double r) : radius(r) {}


    double area() override {

        return 3.14159 * radius * radius;

    }

};


int main() {

    Circle circle(5.0);

    Shape* shape = &circle;
    std::cout << "Area of circle: " << shape->area() << std::endl;


    return 0;
}
```

Question 6: Describe the relationship between the Shape and Circle classes. How does function overriding work in this program?

Answer 6: The Circle class is derived from the Shape class using inheritance. The Shape class has a virtual function area() that is overridden in the Circle class. When a Circle object's area() method is called, the overridden version in the Circle class is invoked, calculating the area of the circle using its radius.

Program 7: Template Function

```
#include <iostream>
template <typename T>
T max(T a, T b) {
    return (a > b) ? a : b;
}


int main() {
    int intMax = max(10, 20);
    double doubleMax = max(3.14, 2.71);


    std::cout << "Max int value: " << intMax << std::endl;
    std::cout << "Max double value: " << doubleMax << std::endl;
    return 0;

}
```

Question 7: Explain the purpose of the max template function. How is it used to find the maximum of two values of different data types?

Answer 7: The max template function is designed to find the maximum of two values of any data type. It uses a template parameter T to handle various data types. In the program, the function is used to find the maximum of two integers and two doubles. The function works for different data types because the greater-than (>) operator is defined for those types.

Program 8: Exception Handling

```
#include <iostream>



int main() {

    int dividend, divisor;



    std::cout << "Enter dividend: ";

    std::cin >> dividend;



    std::cout << "Enter divisor: ";

    std::cin >> divisor;



    try {

        if (divisor == 0) {

            throw "Division by zero is not allowed";
        }

        double result = static_cast<double>(dividend) / divisor;

        std::cout << "Result: " << result << std::endl;

    }

    catch (const char* error) {

        std::cerr << "Error: " << error << std::endl;

    }



    return 0;

}
```

Question 8: Describe the purpose of the try and catch blocks in the program. When and why is an exception thrown and caught?

Answer 8: The try block contains code that might throw an exception. In this program, an exception is thrown if the user attempts to divide by zero. The catch block is used to handle the exception that was thrown. If an exception occurs, the program jumps to the catch block and displays an error message. Exception handling allows the program to gracefully handle unexpected situations and prevent crashes.

Program 9: File Input and Output

```
#include <iostream>

#include <fstream>



int main() {

    std::ofstream outFile("output.txt");

    if (!outFile) {

        std::cerr << "Error opening file" << std::endl;
        return 1;
    }

    outFile << "Hello, file output!" << std::endl;
    outFile.close();

    std::ifstream inFile("output.txt");
    if (!inFile) {
        std::cerr << "Error opening file" << std::endl;
        return 1;
    }

    std::string line;
    while (std::getline(inFile, line)) {
        std::cout << line << std::endl;
    }
    inFile.close();

    return 0;
}
```

Question 9: Explain the purpose of the ofstream and ifstream classes. How is file output and input achieved in the program?

Answer 9: The ofstream class is used for writing data to files, and the ifstream class is used for reading data from files. In the program, an output file stream (outFile) is opened to write "Hello, file output!" to a file named "output.txt". Then, an input file stream (inFile) is opened to read the content from "output.txt" and display it on the console.

Program 10: Dynamic Memory Allocation

```
#include <iostream>



int main() {

   int* ptr = new int;

   *ptr = 42;



   std::cout << "Value: " << *ptr << std::endl;



   delete ptr;



   return 0;

}
```

Question 10: Describe the process of dynamic memory allocation in the program. Why is the delete statement used?

Answer 10: Dynamic memory allocation is achieved using the new operator, which allocates memory on the heap for an int. The allocated memory is accessed using the pointer ptr, and the value 42 is stored there. The delete statement is used to deallocate the memory allocated by new, preventing memory leaks.

Program 11: Operator Overloading

```
#include <iostream>
```

```
class Complex {
private:
    double real;
    double imag;

public:
    Complex(double r, double i) : real(r), imag(i) {}

    Complex operator+(const Complex& other) {
        return Complex(real + other.real, imag + other.imag);
    }

    void display() {
        std::cout << real << " + " << imag << "i" << std::endl;
    }
};

int main() {
    Complex c1(2.0, 3.0);
    Complex c2(4.0, 5.0);

    Complex sum = c1 + c2;
    std::cout << "c1: ";

    c1.display();



    std::cout << "c2: ";

    c2.display();



    std::cout << "Sum: ";

    sum.display();



    return 0;

}
```

Question 11: How is the + operator overloaded in the Complex class? How is the operator used to add two Complex objects?

Answer 11: The + operator is overloaded using the operator+ function in the Complex class. It takes another Complex object as a parameter and returns a new Complex object that represents the sum of the two complex numbers. In the program, the + operator is used to add two Complex objects c1 and c2, and the result is stored in the sum variable.

Program 12: Copy Constructor and Deep Copy

```
#include <iostream>

#include <cstring>



class String {

private:
    char* str;


public:
    String(const char* s) {
        str = new char[strlen(s) + 1];
        strcpy(str, s);
    }


    String(const String& other) {
        str = new char[strlen(other.str) + 1];
        strcpy(str, other.str);
    }


    void display() {
        std::cout << str << std::endl;
    }


    ~String() {
        delete[] str;
    }
};


int main() {
    String original("Hello");
    String copy = original;



    std::cout << "Original: ";

    original.display();



    std::cout << "Copy: ";

    copy.display();



    return 0;

}
```

Question 12: Explain the purpose of the copy constructor in the String class. Why is a deep copy necessary for the str attribute?

Answer 12: The copy constructor is used to create a new object that is a copy of an existing object. In the String class, the copy constructor creates a new dynamic memory block for the str attribute and copies the contents of the original object's str. A deep copy is necessary to ensure that each String object has its own separate memory space for the str attribute. If a shallow copy were used, both objects would point to the same memory, and modifying one object's str would affect the other object.

Program 13: Unary Operator Overloading

```
#include <iostream>



class Counter {

private:

    int count;


public:
    Counter() : count(0) {}


    Counter operator++() {
        return Counter(++count);
    }


    void display() {
        std::cout << "Count: " << count << std::endl;
    }
};


int main() {
    Counter c;


    ++c;
    c.display();


    return 0;
}
```

Question 13: How is the pre-increment (++) operator overloaded in the Counter class? How is the operator used in the program?

Answer 13: The pre-increment operator (++) is overloaded using the operator++ function in the Counter class. It increments the count attribute and returns a new Counter object. In the program, the pre-increment operator is used with the ++c expression, which increments the count and displays the updated value.

Program 14: Subscript Operator Overloading

```
#include <iostream>


class Array {
private:
    int* data;
    int size;

public:
    Array(int s) : size(s) {
        data = new int[size];
        for (int i = 0; i < size; ++i) {
            data[i] = i + 1;
        }
    }

    int& operator[](int index) {
        return data[index];
    }

    void display() {
        for (int i = 0; i < size; ++i) {
            std::cout << data[i] << " ";
        }
        std::cout << std::endl;
    }

    ~Array() {
        delete[] data;
    }
};

int main() {
    Array arr(5);

    std::cout << "Original array: ";
    arr.display();

    arr[2] = 10;

    std::cout << "Modified array: ";
    arr.display();

    return 0;
}
```

Question 14: How is the subscript ([]) operator overloaded in the Array class? How is it used to access and modify elements in the array?

Answer 14: The subscript operator ([]) is overloaded using the operator[] function in the Array class. It takes an integer index and returns a reference to the element at that index in the data array. In the program, the subscript operator is used to access and modify elements in the Array object arr. For example, arr[2] accesses and modifies the element at index 2.

Program 15: Virtual Functions and Polymorphism

```
#include <iostream>



class Shape {

public:

   virtual void draw() {

       std::cout << "Drawing a shape" << std::endl;

   }

};



class Circle : public Shape {

public:

   void draw() override {

       std::cout << "Drawing a circle" << std::endl;

   }

};



class Square : public Shape {

public:

   void draw() override {
       std::cout << "Drawing a square" << std::endl;
   }
};

int main() {
   Shape* shapes[3];
   shapes[0] = new Shape();
   shapes[1] = new Circle();
   shapes[2] = new Square();

   for (int i = 0; i < 3; ++i) {
       shapes[i]->draw();
       delete shapes[i];
   }

   return 0;
}
```

Question 15: Explain the concept of polymorphism demonstrated in this program. How are virtual functions used to achieve polymorphic behavior?

Answer 15: Polymorphism allows objects of different classes to be treated as objects of a common base class, enabling dynamic method binding. In this program, polymorphism is achieved using a base class Shape and derived classes Circle and Square. The draw function in the Shape class is declared as virtual. When the function is called on an object through a base class pointer, the appropriate version of the function in the derived class is executed, based on the actual object's type.

Program 16: Abstract Classes and Pure Virtual Functions

```
#include <iostream>


class Shape {

public:

    virtual double area() = 0;

};


class Circle : public Shape {

private:

    double radius;


public:

    Circle(double r) : radius(r) {}


    double area() override {

        return 3.14159 * radius * radius;

    }

};


class Rectangle : public Shape {

private:

    double width;

    double height;
public:
    Rectangle(double w, double h) : width(w), height(h) {}

    double area() override {
        return width * height;
    }
};

int main() {
    Circle circle(5.0);
    Rectangle rectangle(4.0, 6.0);

    Shape* shapes[2];
    shapes[0] = &circle;
    shapes[1] = &rectangle;

    for (int i = 0; i < 2; ++i) {
        std::cout << "Area: " << shapes[i]->area() << std::endl;
    }

    return 0;
}
```

Question 16: Define an abstract class named Shape with a pure virtual function area(). How are derived classes Circle and Rectangle used to implement the area() function?

Answer 16: The Shape class is defined as an abstract class with a pure virtual function area(), which has no implementation in the base class. The derived classes Circle and Rectangle provide their own implementations of the area() function. When an object of a derived class is accessed through a base class pointer, the appropriate version of the area() function in the derived class is invoked, enabling polymorphism.

Program 17: Multiple Inheritance

```
#include <iostream>



class Shape {

public:

   virtual void draw() {

       std::cout << "Drawing a shape" << std::endl;

   }

};



class Color {

public:

   virtual void fill() {

       std::cout << "Filling with color" << std::endl;

   }

};
```

```
class Square : public Shape, public Color {

public:

    void draw() override {

        std::cout << "Drawing a square" << std::endl;

    }



    void fill() override {

        std::cout << "Filling square with color" << std::endl;

    }

};



int main() {

    Square square;

    square.draw();

    square.fill();



    return 0;

}
```

Question 17: Describe the concept of multiple inheritance demonstrated in this program. How are virtual functions used to resolve ambiguity in the derived class?

Answer 17: Multiple inheritance occurs when a class inherits from more than one base class. In this program, the Square class inherits from both Shape and Color. When there's ambiguity due to the presence of multiple base classes with the same function name, virtual functions are used to resolve it. By declaring the functions draw and fill as virtual in the base classes (Shape and Color), the derived class Square provides specific implementations for these functions. This allows the Square class to exhibit polymorphic behavior.

Program 18: Standard Template Library (STL) Vector

```
#include <iostream>

#include <vector>



int main() {

   std::vector<int> numbers;



   numbers.push_back(10);

   numbers.push_back(20);

   numbers.push_back(30);



   std::cout << "Numbers: ";

   for (int num : numbers) {

      std::cout << num << " ";

   }

   std::cout << std::endl;



   return 0;

}
```

Question 18: Explain the purpose of the std::vector class from the Standard Template Library (STL). How is it used to store and display a collection of integers?

Answer 18: The std::vector class is a dynamic array provided by the STL that can dynamically grow or shrink in size. It is used to store a collection of elements, such as integers in this case. The push_back function adds elements to the end of the vector, and the for loop is used to iterate through the vector and display its contents.

Program 19: Standard Template Library (STL) Map

```
#include <iostream>

#include <map>



int main() {

   std::map<std::string, int> ages;



   ages["Alice"] = 25;

   ages["Bob"] = 30;

   ages["Charlie"] = 28;



   std::cout << "Age of Alice: " << ages["Alice"] << std::endl;



   return 0;

}
```

Question 19: Describe the purpose of the std::map class from the Standard Template Library (STL). How is it used to store and retrieve key-value pairs?

Answer 19: The std::map class is an associative container provided by the STL that stores key-value pairs. It is implemented as a binary search tree, allowing efficient insertion, deletion, and retrieval of elements based on keys. In this program, the ages map stores the ages of individuals with their names as keys. The ages["Alice"] expression retrieves the age associated with the key "Alice".

Program 20: Complex Application - Library System

```
#include <iostream>

#include <vector>

#include <string>



class Book {

private:

    std::string title;

    std::string author;

    bool borrowed;



public:

    Book(std::string t, std::string a) : title(t), author(a), borrowed(false) {}



    void borrow() {

        borrowed = true;

    }



    void returnBook() {

        borrowed = false;

    }



    void display() {

        std::cout << "Title: " << title << std::endl;
        std::cout << "Author: " << author << std::endl;

        std::cout << "Status: " << (borrowed ? "Borrowed" : "Available") << std::endl;

    }

};


class Library {

private:

    std::vector<Book> books;


public:

    void addBook(const Book& book) {

        books.push_back(book);

    }


    void displayBooks() {

        std::cout << "Library Catalog:" << std::endl;

        for (const Book& book : books) {

            book.display();

            std::cout << std::endl;

        }

    }

};


int main() {
    Library library;


    Book book1("The Great Gatsby", "F. Scott Fitzgerald");

    Book book2("To Kill a Mockingbird", "Harper Lee");


    library.addBook(book1);

    library.addBook(book2);


    book1.borrow();


    library.displayBooks();


    return 0;

}
```

Question 20: Describe the complex application demonstrated in this program. How are the Book and Library classes used to model a library system?

Answer 20: This program simulates a library system using the Book and Library classes. The Book class represents a book with attributes such as title, author, and borrowing status. It has methods to borrow and return books, as well as display book details. The Library class manages a collection of Book objects and provides methods to add books and display the library catalog. In the program, two books are added to the library, one is borrowed, and then the library catalog is displayed, showing book details and their availability status. This example showcases the usage of classes, methods, vectors, and modeling of a real-world scenario.

Program 21: Handling Input Validation Exception

```
#include <iostream>

#include <stdexcept>



int main() {

    try {

        int age;



        std::cout << "Enter your age: ";

        std::cin >> age;



        if (age < 0 || age > 120) {

            throw std::invalid_argument("Invalid age value");

        }



        std::cout << "Your age is: " << age << std::endl;

    }

    catch (const std::exception& e) {

        std::cerr << "Error: " << e.what() << std::endl;

    }



    return 0;

}
```

Question 1: What is the purpose of the std::invalid_argument class from the <stdexcept> header in this program?

Question 2: What is the significance of the e.what() function call inside the catch block?

Question 3: How does the program ensure that the user enters a valid age value?

Question 4: What type of exception is caught in the catch block?

Question 5: If the user enters an age value that is not between 0 and 120, how does the program respond?

Program 22: Custom Exception Class

```
#include <iostream>

#include <stdexcept>



class CustomException : public std::exception {

public:

    const char* what() const noexcept override {

        return "This is a custom exception.";

    }

};



int main() {

    try {
        int x;


        std::cout << "Enter a positive integer: ";

        std::cin >> x;


        if (x <= 0) {

            throw CustomException();

        }


        std::cout << "You entered: " << x << std::endl;

    }

    catch (const CustomException& e) {

        std::cerr << "Custom Exception: " << e.what() << std::endl;

    }


    return 0;

}
```

Questions:

Question 1: Explain the purpose of the CustomException class and its relationship to the std::exception class.

Question 2: What is the purpose of the what() function in the CustomException class?

Question 3: How is the CustomException class used to handle invalid input in the program?

Question 4: What is the significance of the const CustomException& parameter in the catch block?

Question 5: How could you modify the CustomException class to provide more specific information about the nature of the exception?

Feel free to use these questions to test your understanding of exception handling in C++ and how it applies to different scenarios.