Assignment 2 (Theory and Practical)

1. What is the purpose of the main function in a C++ program?
2. Explain the significance of the return type of the main function.
3. What are the two valid signatures of the main function in C++?
4. What is function prototyping and why is it necessary in C++?
5. How do you declare a function prototype for a function that returns an integer and takes two integer parameters?
6. What happens if a function is used before it is prototyped?
7. What is the difference between a declaration and a definition of a function?
8. How do you call a simple function that takes no parameters and returns void?
9. Explain the concept of "scope" in the context of functions.
10. What is call by reference in C++?
11. How does call by reference differ from call by value?
12. Provide an example of a function that uses call by reference to swap two integers.
13. What is an inline function in C++?
14. How do inline functions improve performance?
15. Explain the syntax for declaring an inline function.
16. What are macros in C++ and how are they different from inline functions?
17. Explain the advantages and disadvantages of using macros over inline functions.
18. Provide an example to illustrate the differences between macros and inline functions.
19. What is function overloading in C++?
20. How does the compiler differentiate between overloaded functions?
21. Provide an example of overloaded functions in C++.
22. What are default arguments in C++?
23. How do you specify default arguments in a function declaration?
24. What are the rules for using default arguments in functions?
25. Provide an example of a function with default arguments.
26. Write a C++ program that prints "Hello, World!" using the main function.
27. Create a program that takes two integers as command line arguments and prints their sum.
28. Write a function prototype for a function that calculates the factorial of a number.

29. Implement a C++ program with a function prototype for a function that returns the maximum of three numbers.
30. Write a simple function that calculates the area of a circle.
31. Implement a function that takes two integers and returns their product.
32. Create a function that prints the elements of an array.
33. Write a function to swap two integers using call by reference.
34. Implement a function that increments the value of an integer by 10 using call by reference.
35. Create a function that modifies the elements of an array using call by reference.
36. Write an inline function that calculates the square of a number.
37. Create an inline function that returns the cube of a number.
38. Implement a program that uses an inline function to calculate the sum of two numbers.
39. Write a macro to calculate the square of a number and compare its performance with an inline function.
40. Implement a macro to find the maximum of two numbers and compare it with an inline function.
41. Write overloaded functions to calculate the area of a circle, rectangle, and triangle.
42. Implement overloaded functions to find the maximum of two and three numbers.
43. Create overloaded functions to print different data types (int, float, string).
44. Write a function with default arguments to calculate the compound interest.
45. Implement a function with default arguments to print a greeting message (default name is "Guest").
46. Create a function that calculates the power of a number with a default exponent of 2.
47. Write a program to demonstrate the concept of recursive functions.
48. Implement a program that uses an array of function pointers.
49. Create a program to demonstrate the use of function templates.
50. Write a program to illustrate the concept of function pointers and callback functions.

Assignment 3 (Theory and Practical)

1. What is an object in C++?
2. What is a class in C++ and how does it differ from an object?
3. Explain the concept of encapsulation with an example.
4. How do you define a class in C++?
5. Describe the syntax for creating an object of a class.
6. What are private members in a class and how are they accessed?
7. What are public members in a class and how are they accessed?
8. Explain the significance of access specifiers in a class.
9. Provide an example of a class with both private and public members.
10. How does data hiding work in C++?
11. What is a static data member in C++?
12. How do you declare and initialize a static data member?
13. What is a static function member in C++?
14. How do static function members differ from regular function members?
15. Provide an example of a class with static data and function members.
16. What is a constructor in C++ and why is it important?
17. Explain the different types of constructors in C++.
18. What is a default constructor and when is it used?
19. How do parameterized constructors work?
20. What is a copy constructor and what is its purpose?
21. Explain the concept of constructor overloading.
22. How does a constructor initializer list work?
23. What is a destructor in C++ and what is its purpose?
24. How is a destructor declared and defined?
25. What happens if a destructor is not explicitly defined in a class?
26. Explain the concept of automatic and dynamic storage duration in relation to destructors.
27. How do destructors differ from constructors?
28. What is operator overloading in C++ and why is it useful?
29. Describe the syntax for overloading an operator.
30. Which operators can and cannot be overloaded in C++?

31. Provide an example of overloading the "+" operator for a custom class.
32. Explain the concept of friend functions in the context of operator overloading.
33. What is a friend function in C++ and how is it declared?
34. How do friend functions differ from member functions?
35. Explain the benefits and potential drawbacks of using friend functions.
36. What is inheritance in C++ and why is it important?
37. Explain the different types of inheritance in C++.
38. How do you implement single inheritance in C++?
39. What is multiple inheritance and how does it differ from single inheritance?
40. Describe hierarchical inheritance with an example.
41. What is multilevel inheritance and how is it implemented in C++?
42. Explain the concept of hybrid inheritance.
43. What are access modifiers in C++ and what are the different types?
44. How do public, private, and protected access modifiers affect inheritance?
45. Explain how access modifiers control member accessibility in derived classes.
46. What is function overriding in the context of inheritance?
47. How do you override a base class function in a derived class?
48. Explain the use of the "virtual" keyword in function overriding.
49. What is the significance of the "override" specifier in C++11 and later?
50. What is a virtual base class in C++ and why is it used?
51. How do you declare and implement a virtual base class?
52. Explain the role of virtual base classes in resolving ambiguity in multiple inheritance.
53. Provide an example of using a virtual base class to avoid the diamond problem in inheritance.
54. Create a class Person with private attributes name, age, and public methods to set and get the values.
55. Implement a class Student that inherits from Person and adds a private attribute studentID with appropriate methods.
56. Design a class Car with attributes for make, model, and year. Include methods to display car details.
57. Write a program that creates an array of Car objects and displays their details.
58. Implement a class BankAccount with private attributes for account number, balance, and public methods for depositing and withdrawing money.

59. Create a class Rectangle with private attributes for length and width, and public methods to calculate area and perimeter.
60. Write a class Employee with private attributes name, position, and salary, and public methods to display employee details.
61. Create a class Counter with a static data member to count the number of objects created.
62. Implement a class Math with static function members for basic arithmetic operations.
63. Write a class Student with a static data member to keep track of the total number of students enrolled.
64. Implement a class Book with a parameterized constructor to initialize book details.
65. Create a class Point with a default constructor, parameterized constructor, and copy constructor.
66. Write a class Matrix with a parameterized constructor to initialize a 2D array.
67. Implement a class FileHandler with a destructor that closes an open file.
68. Create a class DynamicArray with a destructor that deallocates dynamically allocated memory.
69. Write a class Logger with a destructor that logs messages when the object is destroyed.
70. Overload the "+" operator for a class Complex to add two complex numbers.
71. Implement the "<<" and ">>" operators for a class Fraction to input and output fraction values.
72. Overload the "==" operator for a class Date to compare two dates.
73. Write a class Vector and overload the "[]" operator to access elements of the vector.
74. Implement a class Box with a friend function to calculate the volume of two boxes.
75. Create a class Circle with a friend function to calculate the area.
76. Write a class Distance with a friend function to add two distances.
77. Implement a class Shape with derived classes Circle, Rectangle, and Triangle.
78. Create a class Animal with derived classes Dog, Cat, and Bird.
79. Write a class Vehicle with derived classes Car and Bike.
80. Implement single inheritance with a base class Person and derived class Employee.
81. Create a class Parent and implement multiple inheritance with derived classes Child1 and Child2.
82. Write a class Base and implement hierarchical inheritance with derived classes Derived1, Derived2, and Derived3.
83. Implement multilevel inheritance with classes Base, Intermediate, and Derived.

84. Create a class Base and implement hybrid inheritance with derived classes Derived1, Derived2, and Derived3.
85. Implement a class Library with private, protected, and public members and demonstrate their accessibility.
86. Create a class Account with private data members and public methods to access and modify them.
87. Implement function overriding with a base class Shape and derived class Circle.
88. Create a base class Employee and derived class Manager with overridden methods.
89. Implement a virtual base class Entity with derived classes Person and Organization.
90. Write a class Animal and implement a virtual base class to avoid the diamond problem in inheritance.
91. Implement a class Polynomial with member functions to add and multiply polynomials.
92. Create a class SparseMatrix with member functions for matrix addition and multiplication.
93. Write a class Time with member functions to add, subtract, and compare time values.
94. Implement a class BigNumber to handle arithmetic operations on large numbers.
95. Create a class FileCompressor with member functions to compress and decompress files.
96. Write a class Network with member functions to simulate network packet transmission.
97. Implement a class Cache with member functions to store and retrieve cached data.
98. Create a class Game with member functions to simulate a simple game with player actions and scoring.

Assignment 4 (Theory and Practical)

1. What is polymorphism in C++ and why is it important?
2. Explain the concept of compile-time (static) polymorphism with examples.
3. Describe the concept of runtime (dynamic) polymorphism with examples.
4. What is the difference between static and dynamic polymorphism?
5. How is polymorphism implemented in C++?
6. What are pointers in C++ and how do they work?
7. Explain the syntax for declaring and initializing pointers.
8. How do you access the value pointed to by a pointer?
9. Describe the concept of pointer arithmetic.
10. What are the common pitfalls when using pointers?
11. How are pointers used with objects in C++?
12. Explain the process of dynamically allocating objects using pointers.
13. Provide an example of accessing object members using pointers.
14. What is the difference between a pointer to an object and a reference to an object?
15. How do you release dynamically allocated objects in C++?
16. What is the this pointer in C++ and what is its significance?
17. How is the this pointer used in member functions?
18. Explain how the this pointer can be used to return the current object.
19. What is a virtual function in C++ and why is it used?
20. Describe the syntax for declaring a virtual function.
21. Explain the concept of a vtable (virtual table) and its role in virtual functions.
22. What is a pure virtual function and how is it declared?
23. Provide an example of a class with pure virtual functions.
24. What are the implications of having pure virtual functions in a class?
25. How is polymorphism implemented using inheritance and virtual functions?
26. Provide an example of implementing polymorphism with base and derived classes.
27. Explain the concept of late binding in the context of polymorphism.
28. How does the compiler manage polymorphism in C++?
29. What is an abstract class in C++?
30. How do abstract classes differ from regular classes?

31. Explain the role of abstract methods in abstract classes.
32. Provide an example of defining and using an abstract class.
33. What are the benefits of using abstract classes in C++?
34. What is exception handling in C++ and why is it important?
35. Describe the syntax for throwing and catching exceptions in C++.
36. Explain the concept of try, catch, and throw blocks.
37. What is the role of the catch block in exception handling?
38. Provide an example of handling multiple exceptions in C++.
39. How does the throw keyword work in exception handling?
40. What is the purpose of the finally block in exception handling?
41. How do you create custom exception classes in C++?
42. What are templates in C++ and why are they useful?
43. Describe the syntax for defining a function template.
44. Provide an example of a function template that performs a generic operation.
45. What is a class template and how is it different from a function template?
46. Explain the syntax for defining a class template.
47. Provide an example of a class template that implements a generic data structure.
48. How do you instantiate a template class in C++?
49. What are the advantages of using templates over traditional class inheritance?
50. How do templates promote code reusability in C++?
51. Implement a base class Shape with derived classes Circle, Rectangle, and Triangle. Use virtual functions to calculate the area of each shape.
52. Create a base class Animal with a virtual function speak(). Implement derived classes Dog, Cat, and Bird, each overriding the speak() function.
53. Write a program that demonstrates function overriding using a base class Employee and derived classes Manager and Worker.
54. Write a program to demonstrate pointer arithmetic by creating an array and accessing its elements using pointers.
55. Implement a program that dynamically allocates memory for an integer array and initializes it using pointers.
56. Create a program that uses a pointer to swap the values of two variables.
57. Write a program that creates a dynamic object of a class Student and accesses its members using pointers.

58. Implement a program that uses a pointer to an array of objects to store and display details of multiple Book objects.
59. Create a program that demonstrates the use of a pointer to an object in a class member function.
60. Write a class Box with a member function that returns the current object using the this pointer.
61. Implement a program that uses the this pointer to chain member function calls in a class Person.
62. Create a class Counter with a member function that compares two objects using the this pointer.
63. Write a program that uses pure virtual functions to create an abstract class Vehicle with derived classes Car and Bike.
64. Implement a program that demonstrates runtime polymorphism using a virtual function in a base class Shape and derived classes Circle and Square.
65. Create a class Account with a pure virtual function calculateInterest(). Implement derived classes SavingsAccount and CurrentAccount.
66. Write a program that demonstrates polymorphism using a base class Media and derived classes Book and DVD.
67. Implement a class hierarchy with a base class Appliance and derived classes WashingMachine, Refrigerator, and Microwave. Use virtual functions to display the functionality of each appliance.
68. Create a program that uses polymorphism to calculate the area of different geometric shapes using a base class Shape and derived classes Circle and Rectangle.
69. Write an abstract class Employee with pure virtual functions calculateSalary() and displayDetails(). Implement derived classes Manager and Engineer.
70. Implement an abstract class Payment with a pure virtual function processPayment(). Create derived classes CreditCardPayment and DebitCardPayment.
71. Create an abstract class Device with a pure virtual function turnOn(). Implement derived classes Laptop and Smartphone.
72. Write a program that handles division by zero using exception handling.
73. Implement a program that demonstrates the use of multiple catch blocks to handle different types of exceptions.
74. Create a custom exception class InvalidAgeException and use it to handle invalid age input in a program.
75. Write a program that uses exception handling to manage file input/output errors.
76. Implement a program that demonstrates the use of the finally block to release resources in exception handling.

77. Write a function template to perform a linear search on an array of any data type.
78. Implement a class template Stack with member functions to push, pop, and display elements.
79. Create a function template to find the maximum of two values of any data type.
80. Write a class template LinkedList with member functions to insert, delete, and display nodes.
81. Implement a function template to perform bubble sort on an array of any data type.
82. Create a class template Queue with member functions to enqueue, dequeue, and display elements.
83. Write a program that uses polymorphism to create a menu-driven application for managing different types of bank accounts.
84. Implement a program that demonstrates the use of smart pointers for dynamic memory management.
85. Create a program that uses exception handling and templates to implement a safe array class.
86. Write a program that demonstrates the use of virtual inheritance to avoid the diamond problem in multiple inheritance.
87. Implement a class Polynomial with member functions to add and multiply polynomials using operator overloading.
88. Create a program that uses function pointers to implement a callback mechanism.
89. Write a program that uses class templates and exception handling to implement a generic and robust data structure.
90. Implement a program that demonstrates the use of virtual destructors in a class hierarchy.
91. Create a program that uses a function template to perform generic matrix operations (addition, multiplication).
92. Write a program that uses polymorphism to create a plugin system for a software application.
93. Implement a program that uses class templates to create a generic binary tree data structure.
94. Create a program that demonstrates the use of polymorphism to implement a dynamic dispatch mechanism.
95. Write a program that uses smart pointers and templates to implement a memory-efficient and type-safe container.
96. Implement a program that uses virtual functions and inheritance to create a simulation of an ecosystem with different types of animals.

97. Create a program that uses exception handling and function templates to implement a robust mathematical library.
98. Write a program that uses polymorphism to create a flexible and extensible GUI framework.
99. Implement a program that demonstrates the use of virtual functions and templates to create a generic and reusable algorithm library.
100. Create a program that uses polymorphism, templates, and exception handling to implement a comprehensive and type-safe collection framework.

Assignment 5 (Theory and Practical)

1. What are streams in C++ and why are they important?
2. Explain the different types of streams in C++.
3. How do input and output streams differ in C++?
4. Describe the role of the iostream library in C++.
5. What is the difference between a stream and a file stream?
6. What is the purpose of the cin object in C++?
7. How does the cin object handle input operations?
8. What is the purpose of the cout object in C++?
9. How does the cout object handle output operations?
10. Explain the use of the insertion (<<) and extraction (>>) operators in conjunction with cin and cout.
11. What are the main C++ stream classes and their purposes?
12. Explain the hierarchy of C++ stream classes.
13. What is the role of the istream and ostream classes?
14. Describe the functionality of the ifstream and ofstream classes.
15. How do the fstream and stringstream classes differ from other stream classes?
16. What is unformatted I/O in C++?
17. Provide examples of unformatted I/O functions.
18. What is formatted I/O in C++?
19. How do you use manipulators to perform formatted I/O in C++?
20. Explain the difference between unformatted and formatted I/O operations.
21. What are manipulators in C++?
22. How do manipulators modify the behavior of I/O operations?
23. Provide examples of commonly used manipulators in C++.
24. Explain the use of the setw, setprecision, and fixed manipulators.
25. How do you create custom manipulators in C++?
26. What is a file stream in C++ and how is it used?
27. Explain the process of opening and closing files using file streams.
28. Describe the different modes in which a file can be opened.
29. How do you read from and write to files using file streams?

30. Provide an example of using file streams to copy the contents of one file to another.
31. What are the main C++ file stream classes and their purposes?
32. Explain the role of the ifstream, ofstream, and fstream classes.
33. How do you use the ifstream class to read data from a file?
34. How do you use the ofstream class to write data to a file?
35. Describe the functionality of the fstream class for both input and output operations.
36. What are file management functions in C++?
37. How do you use the remove and rename functions to manage files?
38. Explain the purpose of the seekg and seekp functions in file management.
39. Provide examples of using file management functions to manipulate file pointers.
40. What are file modes in C++?
41. Describe the different file modes available in C++.
42. How do you specify a file mode when opening a file?
43. Explain the difference between binary and text file modes.
44. Provide examples of opening files in different modes using file streams.
45. What are binary files in C++ and how do they differ from text files?
46. Explain the process of reading from and writing to binary files.
47. What are random access files in C++?
48. How do you perform random access operations on files?
49. Provide examples of using file streams to implement random access in binary files.
50. Write a program to perform basic input and output using streams (cin and cout).
51. Create a program that reads and displays multiple lines of text using cin and cout.
52. Implement a program that uses streams to read integers from the user and display their sum.
53. Write a program to input and output various data types using cin and cout.
54. Create a program that formats output using manipulators such as setw, setprecision, and fixed.
55. Implement a program that reads user input for name, age, and salary, and then displays the information using formatted output.
56. Write a program to demonstrate the use of ifstream and ofstream for file input and output.
57. Implement a program that reads a list of integers from a file and displays them on the console.

58. Create a program that writes a list of strings to a file.
59. Write a program to demonstrate unformatted input and output using get and put functions.
60. Implement a program that reads and writes characters using get and put.
61. Create a program that uses formatted input and output to display a table of data.
62. Write a program that uses getline to read a full line of text and display it.
63. Write a program that uses manipulators to format floating-point numbers with different precisions.
64. Implement a program that uses setw to align text output in columns.
65. Create a program that uses manipulators to format currency and percentage values.
66. Write a program to read data from a text file and display it on the console.
67. Implement a program to write user input to a text file.
68. Create a program that copies the contents of one file to another using file streams.
69. Write a program that appends new data to an existing file.
70. Write a program to read binary data from a file using ifstream.
71. Implement a program to write binary data to a file using ofstream.
72. Create a program that demonstrates the use of fstream for both input and output operations.
73. Write a program to read and write complex data structures to a file using binary file streams.
74. Write a program to rename and delete files using the rename and remove functions.
75. Implement a program to create, open, and close files using file streams.
76. Create a program that uses the seekg and tellg functions to manipulate file pointers.
77. Write a program that uses the seekp and tellp functions to set and retrieve the put pointer position.
78. Write a program to open a file in different modes (read, write, append) and demonstrate their effects.
79. Implement a program that reads from and writes to a file in binary mode.
80. Create a program that demonstrates the difference between text and binary file modes.
81. Write a program to open a file in truncation mode and demonstrate its effect.
82. Write a program to read and write binary data to a file using the read and write functions.
83. Implement a program that uses random access to read and write data at specific positions in a binary file.

84. Create a program that reads and writes a structure to a binary file using random access.
85. Write a program that updates specific records in a binary file using random access.
86. Implement a program that reads and displays the contents of a binary file in reverse order.
87. Write a program that uses streams to read user input, process it, and write the results to a file.
88. Implement a program that reads a configuration file and uses its settings to control program behavior.
89. Create a program that logs error messages to a file using file streams.
90. Write a program that uses file streams to create a simple text editor.
91. Implement a program that reads and processes a CSV file using file streams.
92. Create a program that uses file streams to search for a specific word in a text file and count its occurrences.
93. Write a program that demonstrates the use of exception handling with file operations.
94. Implement a program that compresses and decompresses text files using simple encoding techniques.
95. Create a program that uses file streams to merge the contents of multiple text files into a single file.
96. Write a program that reads and processes large data files using memory-mapped files.
97. Implement a program that uses streams to perform basic encryption and decryption of text files.