JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

(40)

B.Tech.-II SEMESTER | T2 EXAM - MARCH 2026

COURSE CODE (CREDITS): 25B11CI211 (4) MAX. MARKS: 25

COURSE NAME: Software Development Fundamentals - II (SDF-II) MAX. TIME: 90 Min.

COURSE INSTRUCTORS: A. Kumar, D. Gupta (Coord.), F. Firdous, H. Srivastava, N. Singla, and P. Aar

***Note:** 1) All questions are compulsory. Marks and COs for each question are indicated. 2) **Answer the questions in the given order.** 3) **Be concise and write neatly.** 4) Use of calculators is not allowed.*

Q. No.	Question	CO	Marks
Q. 1	Explain the different types of inheritance in C++ (single, multiple, multilevel, hierarchical, and hybrid). For each type, highlight its key features, advantages, limitations, and typical use cases. Additionally, discuss how public, protected, and private inheritance affect the accessibility of base class members in derived classes, and how this influences encapsulation and safe code reuse.	2	4
Q. 2	A company is designing an AI-based customer support system using C++ and wants to integrate multiple chatbot models. a) Create an abstract base class Chatbot with a pure virtual function respond(). Derive two classes GPTBot and GeminiBot that override respond() with suitable outputs. b) Demonstrate runtime polymorphism by using an array of base class pointers to invoke different respond() methods. c) Ensure proper memory management by deleting the dynamically allocated objects.	2	4
Q. 3	Write a C++ program to demonstrate exception handling using derived classes in arithmetic operations. Define a base class NumberOperation with a virtual function compute(int a, int b). Derive two classes, Addition and Divide, from the base class. In the Addition class, override compute to perform addition and throw an exception if the result exceeds 100. In the Divide class, override compute to perform division and implement exception handling to catch division by zero. Demonstrate multiple catch clauses and include a catch-all block to handle any unexpected exceptions.	2	4
Q. 4	Consider a '*Library Management System*' in your university, where students borrow and return books, and the system is managed by a librarian. The system involves entities such as Library, Book, Student, Librarian, and Issue Record, capturing details of book issuance and return. Based on this scenario, identify the relevant classes along with their key attributes, and establish appropriate relationships among them using generalization, association, aggregation, and composition. Clearly specify the multiplicities for	5	4

	each relationship and draw a well-structured UML Class Diagram representing the system. Also, briefly justify the choice of relationships used in your design.		
Q. 5	a) Explain the need of pure virtual destructor in C++ and why it must be defined outside the class even though it is declared as pure virtual. b) Briefly explain the diamond problem in multiple inheritance and how ambiguity arises in accessing base class members and constructors. How is this issue resolved in C++? c) Briefly explain how object slicing occurs and how it can be prevented? d) Differentiate between the <<include>> and <<extend>> relationships in a UML use-case diagram. Provide one real-world example of each.	2, 4, 5	[1+2 +1+1 =5]
Q. 6	For each program below, state the output and briefly explain your answer in 1-2 sentences. Assume the following statements are already included: #include <iostream> using namespace std;	4	[1*4 = 4]

a)

```
int main() {
   int temp = 0;
   char* ptr;
   ptr = new char[256];
   try {
      if (temp < 1) { throw temp; }
      if (ptr == NULL) { throw "ptr is NULL"; }}
   catch (const char* p) {
      cout << "Null pointer exception: " << p; }
   catch (...) {
      cout << "Unknown exception occurred"; }
   return 0;
}
```

b)

```
class CTest {
   private:
      mutable int x;
      const int y;
   public:
      CTest() : x(4), y(5) {}
      void modify() const {
         x = 10;}
      void display() const {
         cout << x << " " << y << endl; }
};
int main() {
   const CTest obj;
   obj.modify();
   obj.display();
   return 0; }
```

c)

```
class CBase {
   public:
      virtual void fun() = 0;
};
class CDerived : public CBase {
};
int main() {
   CDerived d;
   return 0;
}
```

d)

```
class A {
   int x;
};
class B {
   int x;
   virtual void fun() {}
};
int main() {
   cout << "Size of class A: " << sizeof(A) << endl;
   cout << "Size of class B (64-bit machine with padding): "
   << sizeof(B);
   return 0; }
```