SECTION A: MULTIPLE CHOICE QUESTIONS **[30 Marks]**

Instruction: Read each question below carefully and choose the letter A, B, C or D that **BEST** describes the answer. Write your answer in the answer booklet. Each question carries **1.5 marks**.

1. In Object-Oriented Programming (OOP), what does "aggregation" refer to?

   A. The process of combining multiple classes into a single class.
   B. A form of association where one class "has-a" relationship with another class.
   C. The process of creating objects from a class.
   D. A type of inheritance where a subclass inherits from multiple parent classes.

2. Which of the following statements indicates the difference between aggregation and composition?

   A. Aggregation is a type of composition, while composition is a type of association.
   B. In aggregation, the enclosing and enclosed objects are independent of each other. While in composition both enclosing and enclosed objects are dependent on each other.
   C. Aggregation describes the “part of” relationship, while composition describes the “has a” relationship between classes.
   D. In composition, when an object is destroyed, other objects that belong to it will not be destroyed. While in aggregation, when an object is destroyed, other objects that belong to it will also be destroyed.

3. Which OOP principles aid in reducing code redundancy from the following selections?

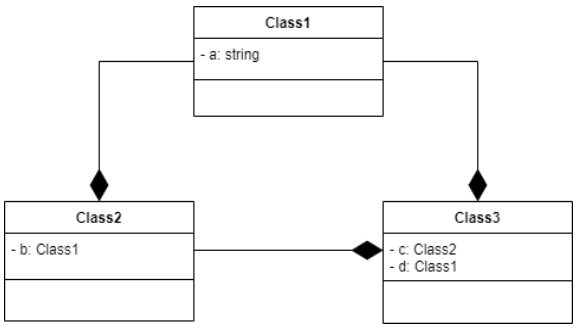
   I. Aggregation
   II. Composition
   III. Inheritance
   IV. Polymorphism

   A. I and II
   B. II and III
   C. III and IV
   D. I, II, III and IV

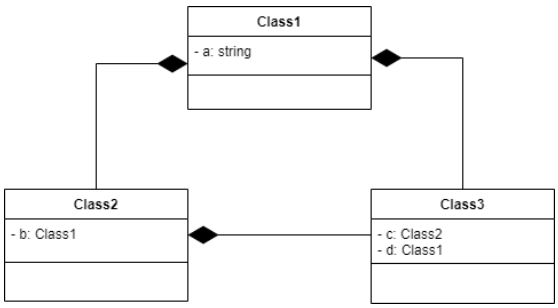
4. Given the class declarations in the C++ code snippet below, which of the following UML diagrams accurately represents the relationship between **`Class1`**, **`Class2`** and **`Class3`**?

```
1   //Class declarations
2   class Class1 {
3     private:
4       string a;
5   };
6
7   class Class2 {
8     private:
9       Class1 b;
10  };
11
12  class Class3 {
13    private:
14      Class2 c;
15      Class1 d;
16  };
```

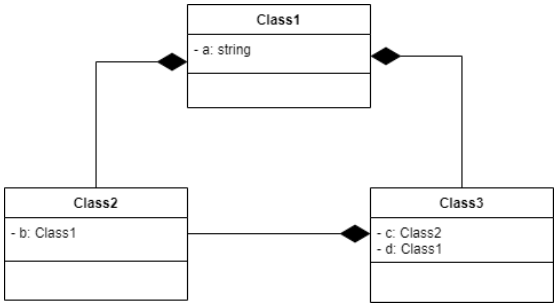
A.



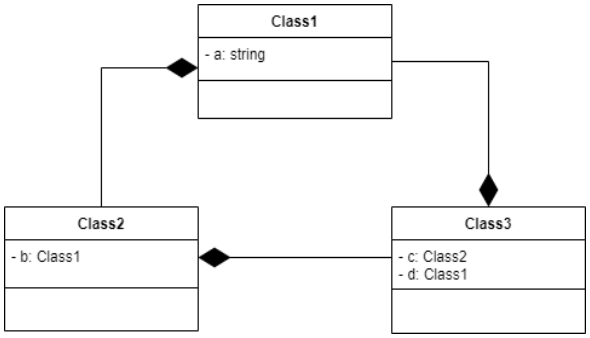
B.



C.



D.



5. Given the C++ code snippet below, what type of relationship(s) is/ are present in the program?

```
1   class Department {
2     //details code are left out for brevity
3   };
4
5   class College {
6     //details code are left out for brevity
7   };
8
9   class Student {
10    // details code are left out for brevity
11    Department dept;
12    College *college;
13  };
```

A. No relationship.

B. Aggregation only.

C. Composition only.

D. Aggregation and composition.

6. If a base class is inherited in protected access mode, which of the following statements is **TRUE**?

A. Only protected members become protected members of the derived class.

B. Only private members of the base class become protected members of the derived class.

C. Public and protected members of the base class become protected members of the derived class.

D. All private, protected, and public members of the base class become private members of the derived class.

7. In object-oriented programming, which statement best characterizes the implications of the inheritance principle on software design?

   A. Inheritance promotes code reuse by allowing derived classes to inherit properties and behaviours from base classes, thus facilitating modularity and extensibility.

   B. Inheritance simplifies program complexity by enforcing a strict hierarchical structure, reducing the need for polymorphism and dynamic binding.

   C. Inheritance inherently leads to tighter coupling between classes, making it challenging to maintain and modify software systems over time.

   D. Inheritance discourages encapsulation by exposing the internal implementation details of base classes to derived classes, leading to potential security vulnerabilities.

8. Given the C++ code snippet below, which of the following statements is **TRUE**?

```
1   class Animal {
2     public:
3       Animal()
4       { cout << "Animal constructor executing" << endl; }
5
6       ~Animal()
7       { cout << "Animal destructor executing." << endl; }
8   };
9
10  class Cat: public Animal {
11    public:
12      Cat(): Animal()
13      { cout << "Cat constructor executing" << endl; }
14
15      ~Cat()
16      { cout << "Cat destructor executing" << endl; }
17  };
18
19  int main() {
20    Animal *myAnimal = new Cat();
21    delete myAnimal;
22    return 0;
23  }
```

   A. In Line 20, both constructors in class Animal and Cat are executed.

   B. In Line 21, both destructors in class Animal and Cat are executed.

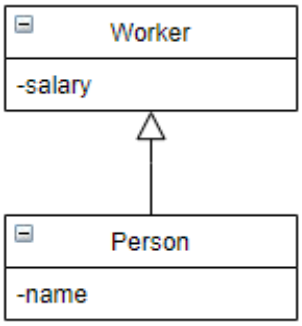
   C. In Line 20, only the constructor in class Animal is executed.

   D. In Line 21, only the destructor in class Cat is executed.

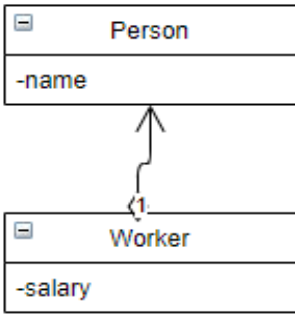
9. Identify the class diagram that corresponds to the C++ code snippet provided below.

```
1   class Worker {
2     double salary;
3   };
4
5   class Person: public Worker {
6     string name;
7   };
```

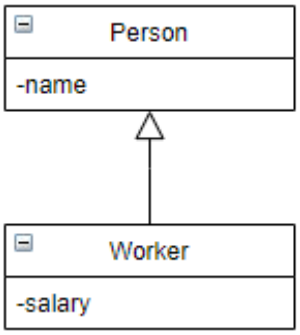
A.



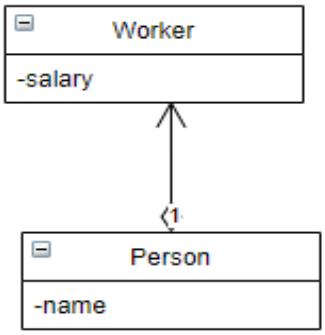
C.



B.



D.



10. Given the two classes (**`Shape`** and **`Square`**) in the C++ code snippet below, which of the following is the correct implementation for the constructor of the class **`Square`**?

```
1   class Shape {
2     private:
3       int color;
4
5     public:
6       Shape(int);
7   };
8
9   Shape::Shape(int _color) { color = _color; }
10
11  class Square: private Shape {
12    private:
13    int size;
14
15    public:
16      Square(int, int);
17  };
```

A.
```
Square::Square(int _color, int _size): Shape(_color) {
  size = _size;
}
```

B.
```
Square::Square(int _color, int _size) {
  Shape(_color);
  size = _size;
}
```

C.
```
Square::Square(int _color, int _size) {
  color = _color;
  size = _size;
}
```

D.
```
Square::Square(int color, int size) {
  this->color = color;
  this->size = size;
}
```

11. Which of the following statements is **TRUE** about a pure virtual method?

    A. A pure virtual method is a method in a parent class declared as virtual but without any definition.

    B. A pure virtual method is a method in a parent class declared as virtual with at least one definition.

    C. Child classes may or may not override pure virtual methods.

    D. A pure virtual method is a method in a child class declared as virtual with at least one definition.

12. What is the primary purpose of polymorphism in object-oriented programming?

    A. Polymorphism allows a derived class to inherit properties and behaviors from a base class.

    B. Polymorphism promotes code organization by grouping related variables and functions together.

    C. Polymorphism enables objects of different classes to be treated as objects of a common base class.

    D. Polymorphism ensures that class members are visible and accessible within a program.

13. Which of the following is the **CORRECT** syntax to declare a pure virtual function named **`display`**?

A. `virtual void display() == 0;`

B. `virtual void display() = 0;`

C. `void virtual display() == 0;`

D. `void virtual display() = 0;`

14. Given the C++ code snippet below, what would be the output of the program?

```
1   class Education {
2     char name[10];
3     public:
4       void disp() {
5         cout << "It's an education system";
6        }
7   };
8
9   class School: public Education {
10    public:
11      void disp() {
12        cout << "It's a school education system";
13      }
14  };
15
16  int main() {
17    Education *e;
18    School s;
19    e = &s;
20    e->disp();
21    return 0;
22  }
```

A. It’s a school education system

B. It’s a school education systemIt’s an education system

C. It’s an education systemIt’s a school education system

D. It’s an education system

15. Given the C++ code snippet below, which sound will be printed when the **`dog.makeSound()`** function is executed in the **`main()`** function?

```
1   class Animal {
2     public:
3       void makeSound() {
4         cout << "Some generic sound\n";
5       }
6   };
7
8   class Dog: public Animal {
```

```
9     public:
10      void makeSound() {
11        cout << "Bark!\n";
12      }
13  };
14
15  int main() {
16    Animal animal;
17    Dog dog;
18    animal.makeSound();
19    dog.makeSound();
20    return 0;
21  }
```

A. Some generic sound

B. Bark!

C. Compilation error

D. Runtime error

16. In C++ programming, several keywords such as **`throw`**, **`try`** and **`catch`** are used to handle exceptions. Which statement is **TRUE** about **`try`**?

A. It sends a signal that an error has occurred.

B. When followed by a block `{}`, it is used to execute code that may throw an exception.

C. When followed by a block `{}`, it is used to detect and handle exceptions thrown in the preceding **`try`** block.

D. It is used as the value that signals an error.

17. Choose a statement that is **TRUE** about error handling with exceptions.

A. Error handling is done by allowing the error to occur and then handling it later by catching it.

B. Error handling is done by anticipating an error before it happens.

C. Error handling is done by safeguarding code with an **`if`** block to prevent any error from occurring.

D. Error handling is done by ignoring the execution of code that may potentially raise errors.

18. Which of the following represents the **CORRECT** syntax for using a function template in C++?

A.
```
template <typename T>
void myFunction(T arg) {
  // Function body
}
```

B.
```
void myFunction<T>(T arg) {
  // Function body
}
```

C.
```
template <typename T>
myFunction(T arg) {
  // Function body
}
```

D.
```
void myFunction(T arg) <typename T> {
  // Function body
}
```

19. Given the C++ code snippet below, what would be the output of the program?

```
1   void divide(int a, int b) {
2     if (b == 0) {
3       throw "Division by zero!";
4     }
5     int result = a / b;
6     cout << "Result of division: " << result << endl;
7   }
8
9   int main() {
10    try {
11      divide(10, 2);
12      divide(8, 0);
13      divide(12, 3);
14    }
15    catch(const char* error) {
16      cout << "Caught an exception: " << error << endl;
17    }
18    return 0;
19  }
```

A. Result of division: 5

Caught an exception: Division by zero!

Result of division: 4

B. Result of division: 5

Result of division: 4

Caught an exception: Division by zero!

C. Caught an exception: Division by zero!
Result of division: 5
Result of division: 4

D. Result of division: 5
Caught an exception: Division by zero!

20. Given the C++ program below, which of the following is the correct way to rewrite the loop at **Line 11** and **Line 12** using an iterator?

```
1   #include <vector>
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6     vector<char> alphabets = {'A', 'B', 'C', 'D'};
7
8     alphabets.pop_back();
9     alphabets.push_back('A');
10
11    for (int i = 0; i < alphabets.size(); i++)
12       cout << alphabets[i];
13
14    return 0;
15  }
```

A. `for (vector<char>::iterator i = vec1.begin(); i != vec1.end(); i++) {cout << vec1[i];}`

B. `for (vector<char>::iterator i = vec1.begin(); i != vec1.end(); i++) {cout << *i;}`

C. `for (vector<char>::iterator i = vec1.begin(); i != vec1.end(); i.next()) {cout << i;}`

D. `for (int i = vec1.size(); i++) {cout << vec1.iterator[i];}`

SECTION B: STRUCTURED QUESTIONS [70 Marks]

Instruction: This section consists of THREE (3) questions. Answer all questions. The marks are as indicated in the question.

1. The question is about object-oriented programming concepts. [25 marks]

   a) Discuss the following concepts and for each of them, give an example use case or scenario.

      i) Access specifiers. (5 marks)

      ii) Overload vs override methods. (5 marks)

      iii) Exceptions. (5 marks)

   b) Explain static binding and dynamic binding in the context of object-oriented programming. (4 marks)

   c) Explain **TWO** differences between composition and aggregation principles in object-oriented programming. Then, explain **ONE** similarity between them? (6 marks)

2. The question is about object-oriented programming implementation in C++ for UML class diagram and output tracing. [20 marks]

   a) Given the C++ code snippet shown in **Figure 1**, draw a UML class diagram that represents all the relationships between the classes. (10 marks)

```
1   class Course {
2     string name;
3
4     public:
5       Course(string _name);
6       String getName() const;
7   };
8
9   class Person {
10    protected:
11      string name;
12      int age;
13
14    public:
15      Person(string _name, int _age);
16      virtual void display() const;
17  };
18
19  class Teacher: public Person {
```

```
20    private:
21      Course *course;
22
23    public:
24      Teacher(string _name, int _age);
25      void display() const;
26  };
27
28  class Student: public Person {
29    private:
30      Course *enrolledCourse;
31
32    public:
33      Student(string_name, int _age);
34      void enrollCourse(Course *course);
35      void display() const;
36  };
37
38  class School {
39    private:
40      Course course1, course 2, course3, course4;
41
42    public:
43      School(string name1, string name2, string name3, string
44      name4): course1(name1), course2(name2), course3(name3),
45      course4(name4);
46      void DisplaySchoolInfo();
47  };
```

Figure 1: C++ Code Snippet for Question 2(a)

b) Given the C++ code snippet shown in Figure 2, what would be the output upon executing the program? (10 marks)

```
1   class Wheel {
2     private:
3       int size;
4
5     public:
6       Wheel(int size) : size(size) {}
7
8       void printWheelDetails() const {
9         cout << "Size: " << size << " inches" << endl; }
10  };
11
12  class Car {
13    private:
14      string name;
15      Wheel wheels[4];
16
17    public:
18      Car(const string& name): name(name), wheels{Wheel(0),
19      Wheel(0), Wheel(0), Wheel(0)} {}
20
21      void addWheel(const Wheel& wheel, int position) {
22        if (position >= 0 && position < 4)
23          wheels[position] = wheel;
24      }
25
```

```
26      void printCarDetails() const {
27        cout << "Car: " << name << endl;
28        for (int i = 0; i < 4; ++i) {
29          cout << "Wheel " << i + 1 << ": ";
30          wheels[i].printWheelDetails(); }
31        cout << endl;
32      }
33  };
34
35  int main() {
36    Car myCar1("Honda");
37    myCar1.addWheel(Wheel(15), 0);
38    myCar1.addWheel(Wheel(16), 1);
39    myCar1.addWheel(Wheel(17), 2);
40    myCar1.addWheel(Wheel(18), 3);
41    myCar1.printCarDetails();
42
43    Car myCar2("Toyota");
44    Wheel wheel1(19);
45    Wheel wheel2(18);
46    Wheel wheel3(17);
47    Wheel wheel4(16);
48
49    myCar2.addWheel(wheel1, 0);
50    myCar2.addWheel(wheel2, 1);
51    myCar2.addWheel(wheel3, 2);
52    myCar2.addWheel(wheel4, 3);
53    myCar2.printCarDetails();
54
55    return 0;
56  }
```

Figure 2: C++ Code Snippet for Question 2(b)

3. The question is about object-oriented programming implementation in C++ for writing code segments and statements. [25 marks]

a) Given the C++ code snippet shown in Figure 3, answer the following questions:

```
1   class Movie {
2     private:
3       string title;
4       string director;
5       int year;
6
7     public:
8       Movie(string t, string d, int year): title(t), director(d),
9       year(year) {
10        cout << "A new movie object has been created.\n";
11        cout << "Title: " << title << ", Director: " << director
12             << ", Year: " << year << '\n';
13      }
14
15      ~Movie() {
16        cout << "The movie object has been destroyed.\n";
17      }
18
19      //a(i) Define a new function named updateMovie
```

```
20      ____________________________________________________________
21      ____________________________________________________________
22  };
23
24  int main() {
25    Movie myMovie("Inception", "Christopher Nolan", 2010);
26    myMovie.updateMovie("Dune: Part Two", "Denis Villeneuve",
27    2024);
28
29    return 0;
30  }
```

Figure 3: C++ Code Snippet for Question 3(a)

i) Based on the output given in **Figure 4** and the function call in the **`main`** function (in **Figure 3**), write a new member function for the **`Movie`** class named **`updateMovie`**. This function should update the details of the **`Movie`** object. Then, the function should display a message indicating that the movie details have been updated as shown in **Figure 4**. (5 marks)

```
A new movie object has been created.
Title: Inception, Director: Christopher Nolan, Year: 2010

Movie details updated.
Title: Dune: Part Two, Director: Denis Villeneuve, Year: 2024
The movie object has been destroyed.
```

Figure 4: Output for code in Figure 3

ii) Modify the **`main`** function to include **`try`** and **`catch`** error handling for the scenario when the user enters an invalid year (a year before 1940 or after 2025). An example of the output of the modified program is shown in **Figure 5**. **Note:** The bold texts indicate input entered by the user. (10 marks)

```
Title: Spider-Man
Director: Sam Raimi
Year: 2026
Error: std::exception
```

Figure 5: Example output for modified program

b) Based on the incomplete C++ program shown in Figure 6, complete the program according to the comments given in the program and the following instructions:

```
1   #include <iostream>
2   #include <string>
3   #include <vector>
4   _________________ //b(i) Include exception header
5
6   using namespace std;
7
8   class Animal {
9     public:
10      string name;
11
12      Animal(string _name) : name(_name) {}
13      virtual void sound() const {
14        cout << name << " makes a sound" << endl; }
15  };
16
17  class Cow: public Animal {
18    public:
19      Cow(string _name) : Animal(_name) {}
20      void sound() const { cout << name << " moo" << endl; }
21  };
22
23  class Cat: public Animal {
24    public:
25      Cat(string _name) : Animal(_name) {}
26      void sound() const { cout << name << " meows" << endl; }
27  };
28
29  //b(ii) Class AnimalNotFoundException to handle exception
30  ____________________________________________________________ {
31    public:
32      const char *what() { return "Animal not found!"; }
33  };
34
35  template <typename T>
36  const T& findAnimal(const vector<T>& animals, const string
37  &name) {
38    for (const auto& animal: animals)
39      if (animal.name == name)
40         return animal;
41
42     //b(iii) Throw the exception class if no animal found
43      ____________________________________________________________
44  }
45
46  int main() {
47    vector<Animal> animals;
48    Cow cow("Charlie");
49    Cat cat("Poppy");
50
51    //Adding animals to the vector
52    animals.push_back(cow);
53    animals.push_back(cat);
54
55    //Accessing and making sounds of animals using the vector
56    //and iterators
57    for (const auto& animal: animals) {
58      //b(iv) Access animal object for its respective sound
59      //function
60      ____________________________________________________________
61    }
```

```
62
63      //b(vi) Find and print the sound of an animal using try and
64      //catch
65      ____________________________________________________________
66
67      return 0;
68  }
```

Figure 6: C++ program for Question 3(b)

i) Write a statement to include the appropriate exception header in the program. (1 mark)

ii) Write a statement to declare the class **`AnimalNotFoundException`** to handle exceptions. (2 marks)

iii) Write a statement to throw the exception defined in b(ii) if no animal is found. (1 mark)

iv) Write a statement to call the **`sound`** function of each **`Animal`** object. (1 mark)

v) Write a code segment to find and print the sound of an animal using **`try`** and **`catch`**. The process is to find if the object **`cat`** with the name "**Poppy"** is found or not. (5 marks)