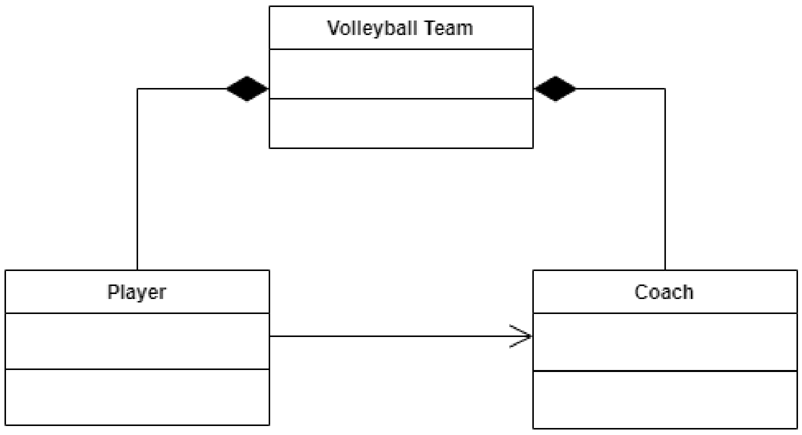
## SECTION A: MULTIPLE CHOICES SINGLE ANSWER QUESTIONS [60 Marks]

**Instruction**: This section consists of FORTY (40) questions. Read each question below carefully and choose the **BEST** answer. Only **ONE** answer is accepted for each question. Each question carries 1.5 marks.

*Note: Questions in Paper 1 will be conducted fully online through E-learning.*



1. Which type of relationship/s is involved in the UML diagram above?
   - Association and composition
   - Association and aggregation
   - Aggregation only
   - Composition only

2. Which one of the following statements is FALSE about composition?
   - Composition is a special form of association, which represents an ownership relationship between two classes.
   - Composition models a whole/part relationship. The part object will be created inside the composed or whole class.
   - In composition, when an object is destroyed, other objects belonging to it will be destroyed as well.
   - The UML class diagram uses a solid diamond attached to the end of the association line to indicate composition.

```
 1  #include <iostream>
 2  using namespace std;
 3  
 4  class Address {
 5          private:
 6                  string street;
 7                  string town;
 8                  string zipcode;
 9          public:
10                  Address (string street, string town, string zipcode);
11                  string getAddress();
12  };
13  
14  Address::Address(string street, string town, string zipcode){
15          this->street=street;
16          this->town=town;
17          this->zipcode=zipcode;
18  }
19  
20  string Address::getAddress()
21  {       return street + "\n" + town + "\n" + zipcode;  }
22  
23  class Resident{
24          private:
25                  string firstname;
26                  string lastname;
27                  Address *home;
28          public:
29                  Resident (string firstname, string lastname, Address
30                  *homeAddress);
31                  void print();
32  };
33  
34  Resident::Resident(string firstname, string lastname, Address *homeAddress){
35          this->firstname=firstname;
36          this->lastname=lastname;
37          home=homeAddress;
38  }
39  
40  void Resident::print(){
41          cout<<firstname<<"\t"<<lastname<<"\n"<<home->getAddress()<<"\n";
42  }
43  
44  int main()
45  {
46      Address *add=new Address("Jalan PI 18/1","Skudai", "81200");
47      Resident res1("Ahmad","Ali",add);
48      res1.print();
49  
50      return 0;
51  }
```

3. Based on Line 27 in the program above, which one of the following classes ia an aggregate class?

   - `Resident`
   - `Address`
   - `Resident` and `Address`
   - `int main ()`

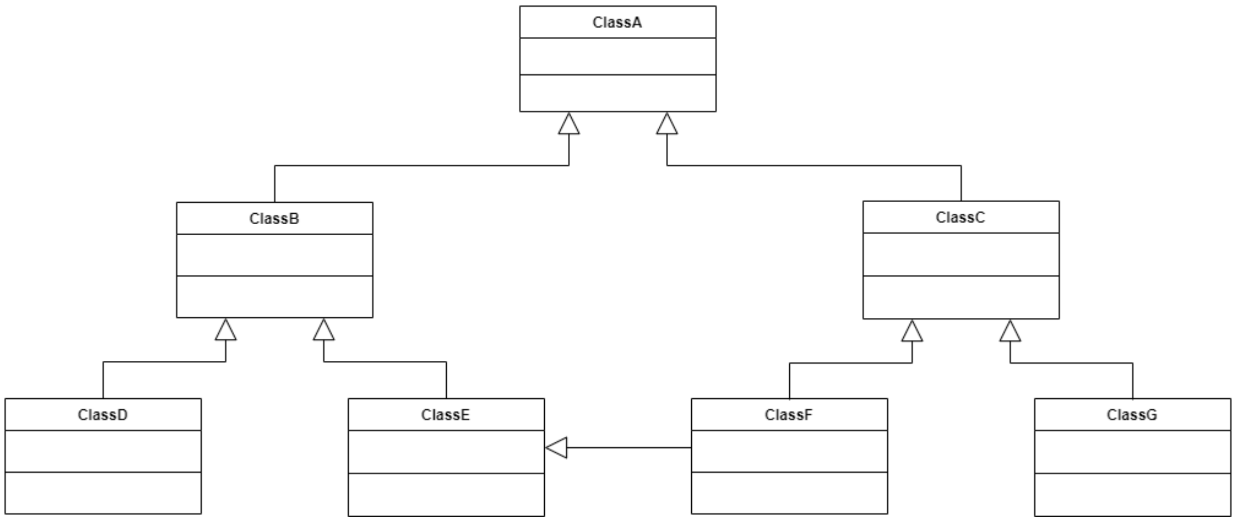
```
//Class declarations

class Company {
        private:
                string company_name;
        protected:
                int num_division;
        public:
                int num_employee;
};

class Division:protected Company
{
        private:
                int division_name;
        protected:
                string function;
        public:
                int num_staff;
};

class Tasks:public Division
{
        private:
                int task_description;
        protected:
                string task_relation;
        public:
                int staff_task;
};
```

4. Based on the class declarations as stated in the program segment above, how do members in class named `Company` appear in the class named `Division`?

| | |
|---|---|
| • | `company_name` is inaccessible<br>`protected: num_division`<br>`protected: num_employee` |
| • | `protected: company_name`<br>`protected: num_division`<br>`protected: num_employee` |
| • | `protected: company_name`<br>`protected: num_division`<br>`public: num_employee` |
| • | `company_name` is inaccessible<br>`num_division` is inaccessible<br>`protected: num_employee` |

5. Which one of the following statements is CORRECT about constructors and destructors in base and derived classes?
   - Both base and derived classes can have their own constructors and destructors.

- When an object of a derived class is created, the derived class's constructor is executed first , followed by the base class's constructor.
- When an object of a derived class is destroyed, the base class's destructor is called first, then that of the derived class.
- Only base classes can have their own constructors and destructor while derived classes initiate their objects through their base classes.



6. Which one of the following statements is CORRECT in describing the above UML diagram?
    - Both `ClassB` and `ClassD` are child classes to `ClassA`.
    - Both `ClassE` and `ClassF` are child classes to `ClassC`.
    - `ClassE` is child class for both `ClassF` and `ClassC`.
    - `ClassF` is child class for `ClassC`, `ClassE` and `ClassG`.

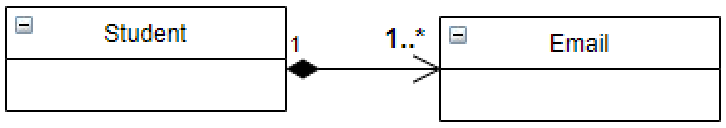
7. Which one of the following statements is CORRECT about an abstract class?
    - An abstract class is a class that cannot have any objects.
    - An abstract class is created when a class contains more than one pure virtual methods.
    - An abstract class can have objects like its child classes.
    - An abstract class can be created although it does not contain pure virtual methods.

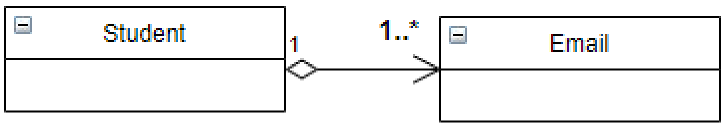
8. Which one of the following statements is TRUE about a pure virtual method?
    - A pure virtual method is a method in a parent class declared as virtual but without any definition.

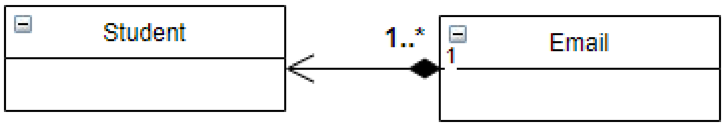
- A pure virtual method is a method in a parent class declared as virtual with at least one definition.
- Child classes may or may not override pure virtual methods.
- A pure virtual method is a method in a child class declared as virtual with at least one definition.

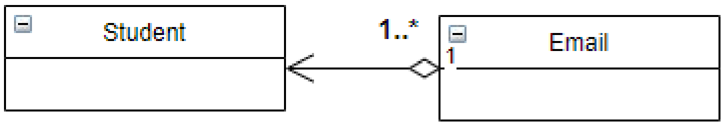
9. There are several keywords such as `throw`, `try` and `catch` used to handle exceptions in C++ programming. Which statement is TRUE about `try`?
   - Used by followed by a block `{}`, is used to invoke code that throws an exception.
   - Used to send a signal that an error has occurred.
   - Used by followed by a block `{}`, is used to detect and process exceptions thrown in preceding try block.
   - Used as the value that signals an error.

10. Which one of the following statements is FALSE about function templates?
    - <u>Function templates cannot be overloaded.</u>
    - Each template must have a unique parameter list.
    - All data types specified in template prefix must be used in template definition.
    - Function calls must pass parameters for all data types specified in the template prefix.

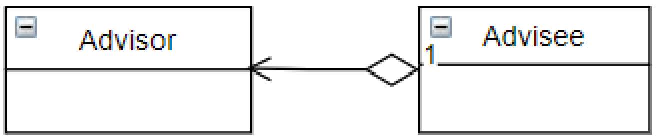
11. Each student in UTM has at least one email address, i.e. the one provided by UTM itself. Most of the students hold other external email accounts such as from gmail.com, microsoft.com, etc. Which of the following class diagrams is the correct way to model this scenario?
    - 

    - 

- 

- 

12. Consider two classes in the following class diagram. What does the diagram tell?



- Each advisee knows who his/her advisor is
- Each advisor knows who his/her advisee is
- Each advisee must have an advisor
- Each advisor must have at least an advisee

```
class Item{
private:
    int data;

public:
    Item(int _data) : data(_data) {}
    int getData() const { return data; }
};

class List{
private:
    Item *items[100];    // array to hold elements
    int count;           // number of elements

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

    // addItem():  To add a new element to the array
    void addItem(Item *item) {
        ____________________
    }
22        // getItem():  To return the element  at  index  i-th
23        Item *getItem(int i){
24            ________________________
25        }
26    };
27
```

13. Consider the two classes in the program segment above, `List` and `Item`, respectively. What should be written in the implementation of the method `addItem()` at Line 20.

- `items[count++]= item;`
- `items[count]= item;`
- `items[count++]= *item;`
- `items[count]= &item;`

```
 1  class Item{
 2  private:
 3      int data;
 4
 5  public:
 6      Item(int _data) : data(_data) {}
 7      int getData() const { return data; }
 8  };
 9
10  class List{
11  private:
12      Item *items[100];    // array to hold elements
13      int count;           // number of elements
14
15  public:
16      List() : count(0) {}
17
18      // addItem():  To add a new element to the array
19      void addItem(Item *item) {
20          ________________________
21      }
22
23      // getItem():  To return the element  at  index  i-th
24      Item *getItem(int i){
25          ________________________
26      }
27  };
```

14. Consider the two classes in the program segment above, `List` and `Item`, respectively. What should be written in the implementation of the method `getItem()` at Line 24.

- `return items[i];`
- `return items + i;`
- `return items;`
- `return *items+i;`

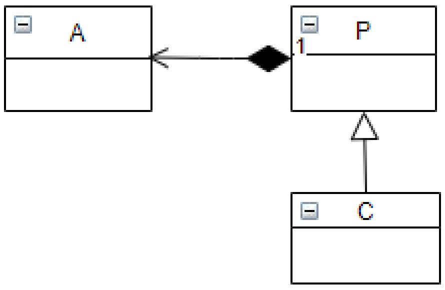
```
class Shape{
protected:
    int color;
};

class Circle : private Shape{
};
```

15. Consider the two classes in the above program. What will be the resulting access specifier for the attribute `color` in the class `Circle`?

- Private
- Protected
- Public
- The class `Circle` cannot access the attribute `color`.

16. All the followings are what the inheritance principle of Object-Oriented Programming offer EXCEPT ________________

- Reducing memory consumption
- Generalization
- Specialization
- Reducing code duplication



17. Based on the class diagram in the above figure, which of the following is TRUE about the relationship between the classes.
    - Class C and A form a composition relationship.
    - Class C and A form a relationship.
    - Class C and P form a composition relationship.
    - Class A and P form an inheritance relationship.

```
class Point{
private:
    int x, y;
};

class Shape{
protected:
    Point location;
    int color;
};

class Circle : public Shape{
private:
    int radius;
};
```

18. Consider the class declarations of three classes in the above program. If an object is created from the class `Circle`, what will the object have?
    - `x, y, color` and `radius`
    - only `radius`
    - `location` and `radius`
    - `x, y, color` and `radius`

19. To implement polymorphism in C++, classes must be arranged in ______________________ relationships.
    - inheritance
    - composition
    - aggregation
    - abstraction

20. What is a pure method in C++?

- a method without a body of definitions.
- a method that does not change any attributes of the class.
- a method that sets the attributes to a zero value.
- a method that is purely written without reference to any other method.

21. Which of the following is FALSE about an abstract class in C++?
    - an abstract class is declared with the virtual keyword.
    - an abstract class cannot instantiate any objects.
    - a class declared with pure methods automatically becomes an abstract class.
    - an abstract can be inherited by other classes.

22. The working principle of class templates in C++ is based on the _______________ concept.
    - generalization
    - inheritance
    - polymorphism
    - association

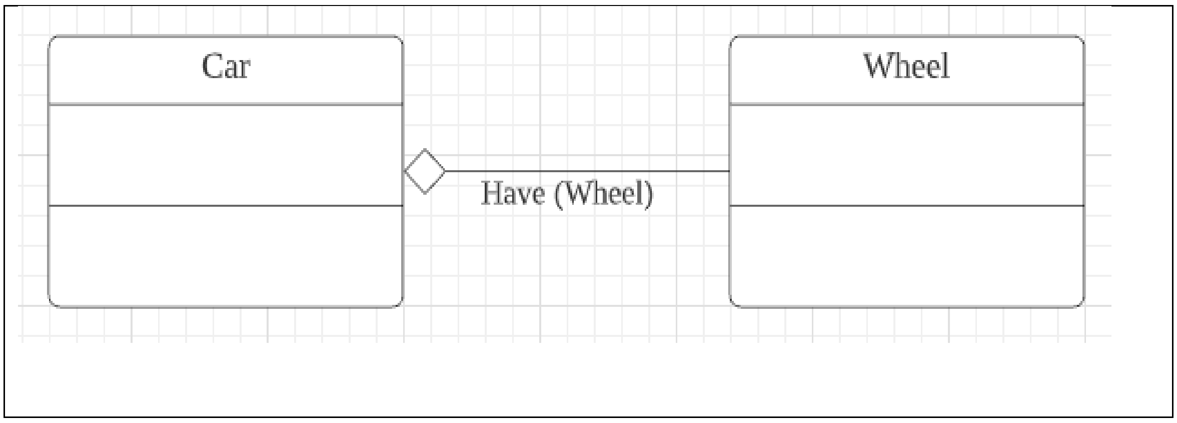
23. All the followings are the keywords for dealing with exceptions in C++ EXCEPT ____
    - `rethrow`
    - `throw`
    - `try`
    - `catch`

24. Choose a statement that is TRUE about error handling with exceptions?
    - Error handling is done by allowing the error to occur then it is handled by catching it later.
    - Error handling is done by anticipating an error before it happens.
    - Error handling is done by safeguarding code with the `if` block to prevent any error to occur.
    - Error handling is done by ignoring the execution of code that may raise errors.

25. A **folder** class that contains a **file** class can be represented as ____________________ relationship.

- composition
- aggregation
- association
- inheritance

26. Consider the class diagram below. Which of the following scenarios is INCORRECT of what the class diagram illustrates?



- A wheel always needs a car.
- Linked objects are not dependent upon the other object.
- The wheel object is meaningful even without the car object.
- Car and wheel relationship is called UML Aggregation relation.

27. Which of the following statement is CORRECT?

- Both aggregation and composition are uni-directional relationships.
- Composition is a weak type of association relationship because objects are strongly independent on each other
- Both aggregations and association of objects of related classes cannot occur independently
- Association models the is-a relationship between classes

```
class Payment
{
  public:
  double basic = 40000;
};

class Loan : public Payment
{
  public:
  double withInterest = 50400;
};

int main()
{
  Loan loan1;
  cout << loan1.basic<<endl;
  cout << loan1.withInterest;
}




```

28. What will be the output of Program A-1 above? Assume that all header file already being coded.

- `40000`
  `50400`
- `50400`
  `40000`
- Compilation error
- Run time error

29. Members which are not intended to be inherited are declared as ________________

- Private members
- Public members
- Protected members
- Private or Protected members

30. Which of the following statements is FALSE about the program below?

```
//Program
class Machine
{
    . . .
};

class Auto:public Machine
{
    . . .
};

class Manual: protected
Machine
{
    . . .
};
```

- A `Auto` object can access all public members defined in `Machine`.
- The base class is `Machine` and the derived class is `Auto` and `Manual`.
- This program applies the concept of inheritance.
- An `Manual` object has all members declared in `Machine`.

31. Which among the following best describes polymorphism?

- It is the ability for a message/data to be processed in more than one form
- It is the ability for a message/data to be processed in only 1 form
- It is the ability for many messages/data to be processed in one way
- It is the ability for undefined message/data to be processed in at least one way

32. Syntax for Pure Virtual Function is _______________ .

- `virtual void display()=0`
- `virtual void display ()==0`
- `void virtual display ()==0`
- `void virtual display ()=0`

```
abstract class Class1
{
   public : int score;
   comp();
}
class Class2:public Class1
{
    public : calc_grade()
    {
        return 30;
    }
};
```

```
class Class3:public Class1
{
      public : comp()
      {
           return 90;
      }
};
class Evaluate{ int score; };


```

33. Which class/set of classes can illustrate polymorphism in program above?

- All class `Class1`, `Class2` and `Class3` together can show polymorphism
- Only class `Class1` can show polymorphism
- Only class `Class1` and `Class2` together can show polymorphism
- Only Class `Evaluate` and `Class1` together can show polymorphism

34. If inner catch handler is not able to handle the exception then__________ .

- Compiler will check for appropriate catch handler of an outer try block
- Compiler will look for outer try handler
- Program terminates abnormally
- Program will be returned to the main function

35. What is the main purpose of templates?

- To create a generic class
- To promote code duplication.
- To produce code reusability.
- To manipulate the class and make the program run faster.

36. Which problem may arise if we use abstract class functions for polymorphism?

- All the derived classes must implement the undefined functions
- All classes are converted as abstract class
- Derived class must be of abstract type
- Derived classes cannot redefine the function

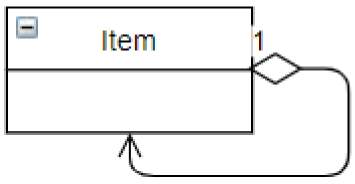
37. What do composition, inheritance and class templates have in common?

- They can reduce code duplication.

- They can reduce memory consumption.
- They allow  a class to have copies of data from other classes.
- They allow an object to share data with other objects.

38. Which one of the statements below indicates the difference between aggregation and composition?
    - Aggregation describes the “have a” relationship while composition describes the “part of” relationship between classes
    - Aggregation is a type of composition while the composition is a type of association.
    - In aggregation, the enclosing and enclosed objects are highly dependent on each other. While in composition both enclosing and enclosed objects are independent.
    - 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 be destroyed as well.

39. What does the following class diagram tell?



    - An item forms an aggregation relationship with another item.
    - An item owns another item.
    - An item is composed from another item.
    - There will be an infinite loop in a method of the class item.

40. How is the relationship is made in **association**?
    - Through the objects of classes
    - Through the constructor of classess
    - Through the default value of the constructor
    - Through the creating of pointer in constructor

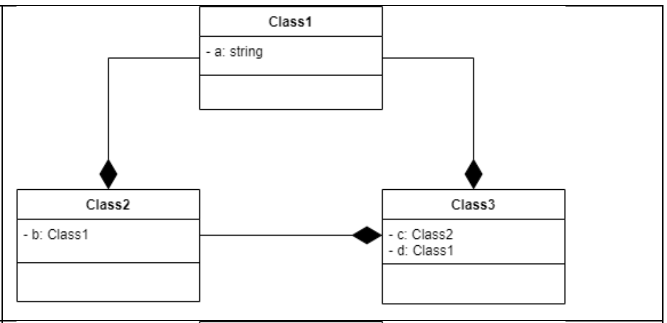
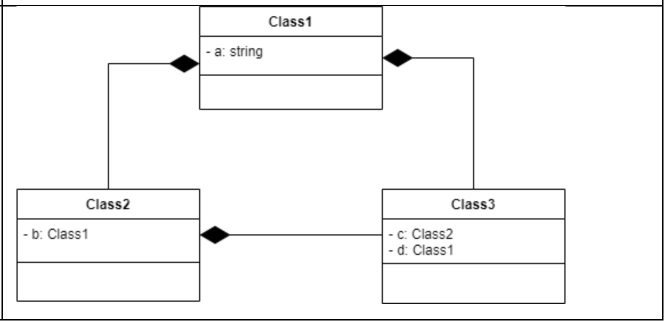
## SECTION B: MULTIPLE CHOICES SINGLE ANSWER QUESTIONS [40 Marks]

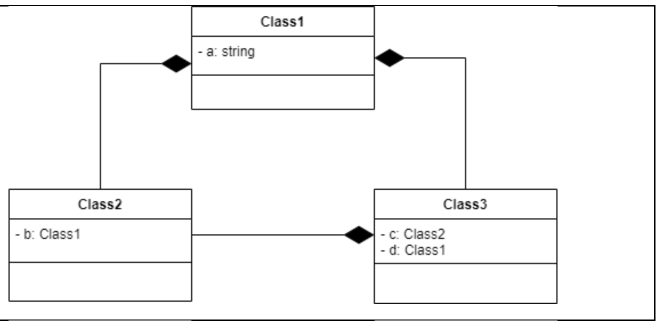
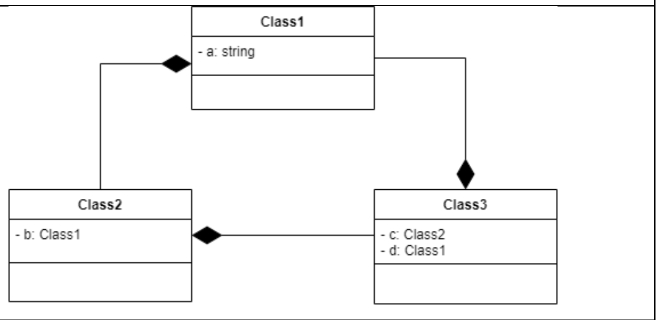
**Instruction:** This section consists of TWENTY (20) questions. Read each question carefully and choose the **BEST** answer. Each question in section A is worth **TWO (2) marks**.

*Note: Questions in Paper 1 will be conducted fully online through E-learning.*

```
//Class declarations

class Class1
{
        private:
                string a;
};

class Class2
{
        private:
                Class1 b;
};

Class Class3
{
        private:
                Class2 c;
                Class1 d;
};
```

41. Given class declarations in the above code segment. Which one of the UML diagrams below indicates the relationship between the class `Class1`, `Class2` and `Class3`.

| | |
|---|---|
| ● |  Class1<br>- a: string<br>Class2<br>- b: Class1<br>Class3<br>- c: Class2<br>- d: Class1 |
| ● |  Class1<br>- a: string<br>Class2<br>- b: Class1<br>Class3<br>- c: Class2<br>- d: Class1 |

|  |  |
|---|---|
| ● |   Class1<br>- a: string<br>Class2<br>- b: Class1<br>Class3<br>- c: Class2<br>- d: Class1 |
| ● |   Class1<br>- a: string<br>Class2<br>- b: Class1<br>Class3<br>- c: Class2<br>- d: Class1 |

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  class Animal
 5  {
 6          public:
 7                  Animal()
 8                  { cout << "Animal constructor executing" << endl; }
 9
10                  ~Animal()
11                  { cout << "Animal destructor executing." << endl; }
12  };
13
14  class Dog : public Animal
15  {
16          public:
17                  Dog() : Animal()
18                  { cout << "Dog constructor executing"  << endl; }
19
20                  ~Dog()
21                  { cout << "Dog destructor executing"  << endl; }
22  };
23
24  int main()
25  {
26
27          Animal *myAnimal = new Dog();
28
29          delete myAnimal;
30
31          return 0;
32  }
```

42. Based on the program above, which one of the statements below is CORRECT?

- In Line 34, both constructors in class `Animal` and `Dog` are executed.
- In Line 36, both destructors in class `Animal` and `Dog` are executed.

- In Line 34, only constructors in class `Animal` is executed.
- In Line 36, only destructors in class `Dog` are executed.

***Taxonomy Level: C3***

```cpp
#include <iostream>
using namespace std;

class Shape
{
	public:
		Shape()
		{ cout << "Shape constructor executing."  << endl; }

		virtual ~Shape()
		{ cout << "Shape destructor executing"  << endl; }
};

class Rectangle : public Shape
{
	public:
		Rectangle() : Shape()
		{ cout << "Rectangle constructor executing"  << endl; }

		~Rectangle()
		{ cout << "Rectangle destructor executing"  << endl; }
};

class Circle : public Shape
{
	public:
		Circle() : Shape()
		{ cout << "Circle constructor executing"  << endl; }

		virtual ~Circle()
		{ cout << "Circle destructor executing"  << endl; }
};


int main()
{
	Shape *myShape1 = new Circle();
	Shape *myShape2 = new Rectangle();

	delete myShape1;

	return 0;
}
```

43. Based on the program above, which output will be displayed after the program execution?

- 
```
Shape constructor executing.
Circle constructor executing.
Shape constructor executing.
Rectangle constructor executing.
Circle destructor executing.
Shape destructor executing.
```

| | |
|---|---|
| ● | `Shape constructor executing.`<br>`Circle constructor executing.`<br>`Shape constructor executing.`<br>`Rectangle constructor executing.` |
| ● | `Shape constructor executing.`<br>`Circle constructor executing.`<br>`Shape constructor executing.`<br>`Rectangle constructor executing.`<br>`Shape destructor executing.` |
| ● | `Shape constructor executing.`<br>`Circle constructor executing.`<br>`Shape constructor executing.`<br>`Rectangle constructor executing.`<br>`Circle destructor executing.` |

```
 1  #include <iostream>
 2  using namespace std;
 3  
 4  class Base
 5  {
 6          public:
 7                  void print() const
 8                  { cout << "This is Base::print()" << endl; }
 9  };
10  
11  class Derived : public Base
12  {
13          public:
14                  void print() const
15                  { cout << "This is Derived::print()" << endl; }
16  };
17  
18  int main()
19  {
20          Base *d = new Derived();
21  
22          d->print();
23          return 0;
24  }
```

44. The program above intends to call `Derived` class twice in Line 26 and Line 27, however the output as shown below is after the program is executed. Which one of the following statements indicates the CORRECT action to fix the error?

```
This is Base::print()
This is Derived::functionA
This is Derived::print()
```

```
This is Base::print()
```

- Change `virtual void functionA(long arg) const` in Line 15 to `virtual void functionA(int arg) const override`
- Change `virtual void functionA(long arg) const` in Line 15 to `virtual void functionA(long arg) const override`
- Change `virtual void functionA(int arg) const` in Line 15 to `virtual void functionA(int arg) const final`
- Change `virtual void functionA(int arg) const` in Line 15 to `virtual void functionA(int arg) const override`

***Taxonomy Level: C3***

```cpp
// This program demonstrates Rectangle class exceptions.
#include <iostream>
#include "Rectangle.h"
using namespace std;

int main()
{
   double width;
   double length;

   Rectangle myRectangle;

   cout << "Enter the rectangle's width: ";
   cin >> width;
   cout << "Enter the rectangle's length: ";
   cin >> length;

try
{
   myRectangle.setWidth(width);
   myRectangle.setLength(length);
   cout << "The area of the rectangle is " << myRectangle.getArea() <<endl;
}

catch (Rectangle::NegativeWidth)
{
   cout<<"Error:A negative value was given "<<"for the rectangle width.\n";
}

catch (Rectangle::NegativeLength)
{
   cout<<"Error:A negative value was given "<<"for the rectangle's length.\n";
}
34
35       cout << "End of the program.\n";
36       return 0;
37    }
```

45. Based on the program above, which one of the following message will be displayed if the user enters `-5` for code in line 14 and `5` for code in line 16?

- `Error: A negative value was given for the rectangle's width.`
- `Error: A negative value was given for the rectangle's length.`
- `The area of the rectangle is -25`
- `The area of the rectangle is 25`

```
class Rectangle
{
   private:
      double width;
      double length;
   public:
      void setData(double w, double l){ width = w; length = l;}
      double getWidth(){ return width; }
      double getLength(){ return length; }
};
```

46. Based on the class declaration above, which one of the following Rectangle class declaration is CORRECT after rewrite as a template that will accept any data type for these members?

-
  ```
  template <class T>
  class Rectangle
  {
     private:
        T width;
        T length;
     public:
        void setData(T w, T l){ width = w; length = l;}
        T getWidth(){ return width; }
        T getLength(){ return length; }
  }
  ```

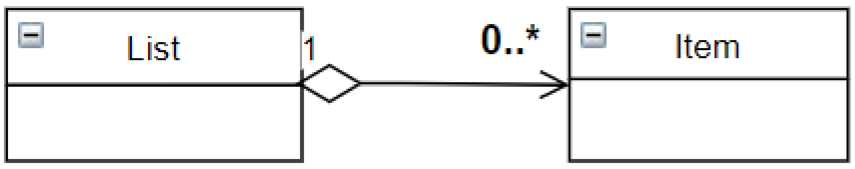
-
  ```
  template <class T>
  class Rectangle
  {
     private:
        int width;
        int length;
     public:
        void setData(T w, T l){ width = w; length = l;}
        T getWidth(){ return width; }
        T getLength(){ return length; }
  };
  ```

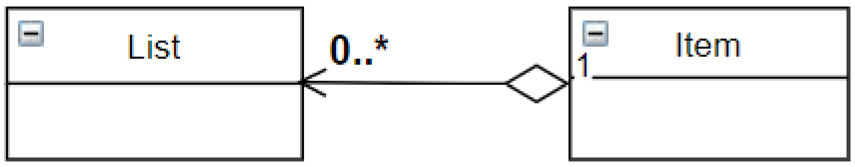
-
  ```
  template <class T>
  class Rectangle
  {
     private:
        T width;
        T length;
     public:
        void setData(double w, double l){ width = w; length = l;}
        double getWidth(){ return width; }
        double getLength(){ return length; }
  };
  ```

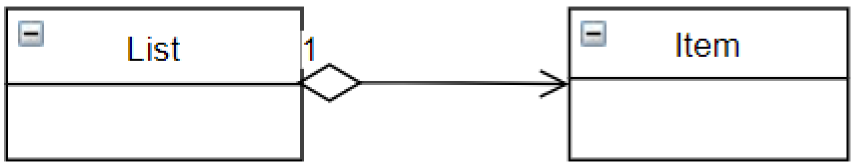
-
  ```
  template <class T>
  class Rectangle
  {
     private:
        int width;
        int length;
     public:
        void setData(int w, int l){ width = w; length = l;}
        int getWidth(){ return width; }
        int getLength(){ return length; }
  };
  ```

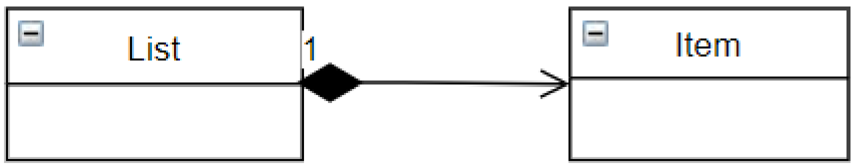
```
class Item{
private:
    int data;

public:
    Item(int _data) : data(_data) {}
    int getData() const { return data; }
};

class List{
private:
    Item *items[100];   // array to hold elements
    int count;          // number of elements

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

    // addItem():  To add a new element to the array
    void addItem(Item *item) {
         ___________________________
    }

    // getItem():  To return the element  at  index  i
    Item *getItem(int i){
        ___________________________
    }
};
```

47. Consider the two classes in the program segment above, `List` and `Item`, respectively. Which one of the following class diagrams is the representation of the relationship between the two classes.

- 

- 

- 

- 

```
class Shape{
private:
    int color;

public:
    Shape(int);
};

Shape::Shape(int _color) { color = _color; }


class Square : private Shape {
private:
    int size;

public:
    Square(int, int);
};
```

48. Consider two classes in the program above, how do you define the constructor of the second class?

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

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

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

-
  ```
  Square::Square(int color, int size){
  ```

```
        this->color = color;

        this->size = size;

    }
```

```
class Data{
private:
    int value;

    bool validate();

public:
    Data(int);
    int getValue() const;
};
```

49. Consider the class `Data` declared in the above program. If the class is inherited by another class with a `private` access specifier, what will the new class inherit from the class?

- the attribute and all the methods.
- only the private attribute and the private method.
- only the private method.
- only the private attribute.

```
class Shape {
private:
    int x, y;

public:
    virtual void reset() = 0;
    virtual int area() const = 0;
};

class Rectangle : public Shape {
protected:
    int width, height;

public:
    int area() const { return width * height; }
};

class Square : public Rectangle {
public:
    void reset() { width = height = 0; }
};
```

50. Consider the above program. Which of the classes are abstract classes (or an abstract class)?

- `Shape` and `Rectangle`
- All the classes
- `Rectangle` and `Square`
- None of the classes is an abstract class

```
#include <string>
using namespace std;

class Person {
private:
    string name;

public:
    Person(string _name="") : name(_name) {}
};

class Worker : public Person {
private:
    double salary;

public:
    Worker(string _name="", double _salary=0.0)
        : salary(_salary), Person(_name) {}

    virtual double overtime() = 0;
};
```

51. Assuming two classes have been defined as in the program above. Based on these classes, all the following declarations are able to compile EXCEPT ______

- `Worker s;`
- `Person p;`
- `Person *q;`
- `Worker *r;`

```cpp
#include <vector>
#include <iostream>
using namespace std;

int main()
{
    vector<char> alphabets = {'A', 'B', 'C', 'D'};

    alphabets.pop_back();
    alphabets.push_back('A');

    for (int i = 0; i < alphabets.size(); i++) {
        cout << alphabets[i];
    }

    return 0;
}
```

52. What would be the output of the above program?

- `ABCA`
- `ABCD`
- `ACBA`
- `DCBA`

```cpp
 1  #include <vector>
 2  #include <iostream>
 3  using namespace std;
 4
 5  int main()
 6  {
 7      vector<char> alphabets = {'A', 'B', 'C', 'D'};
 8
 9      alphabets.pop_back();
10      alphabets.push_back('A');
11
12      for (int i = 0; i < alphabets.size(); i++) {
13          cout << alphabets[i];
14      }
15      return 0;
16  }
```

53. Consider the program above. If you rewrite the loop at Line 12 to Line 14 using an iterator, how would the code look like?

-

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

-

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

-

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

-

```
for ( int i = alphabets.size(); i++ ){
        cout << alphabets.iterator(i);
}
```

```
string doOperation(string s)
{
    string result;

    // assume the rest codes has been written here

    return result;
}

double doOperation(double d)
{
    double result;

    // assume the rest codes has been written here

    return result;
}

void processStringData(string data)
```

```
{
    string result = doOperation(data);
    cout << result << endl;
}

void processDoubleData(double data)
{
    double result = doOperation(data);
    cout << result << endl;
}
```

54. Consider four functions in the above program. How should you rewrite the third and fourth functions, i.e, `processStringData()` and `processDoubleData()` that process different types of data, using a function template? Note that all the overloaded functions for `doOperantions()` should remain unchanged.

-
  ```
  template <class T>
  void processData(T data)
  {
      T result = doOperation(data);
      cout << result << endl;
  }
  ```

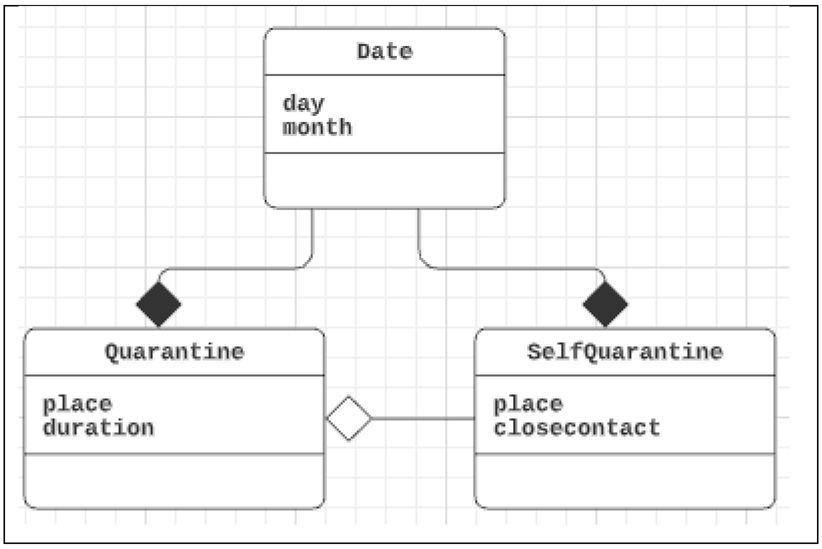
-
  ```
     void processData<Template T>(T data){
      T result = doOperation<T>(data);
      cout << result << endl;
  }
  ```

-
  ```
  template <class T = string>
  ```

```
void processData(T data){
    T result = doOperation(data);
    cout << result << endl;
}

template <class T = double>
void processData(T data){
    T result = doOperation(data);
    cout << result << endl;
}
```

●

```
void processData<T string, double>(T data){
    T result = doOperation(data);
    cout << result << endl;
}
```



55. Consider the class diagram above. Which one of the following code segments is the CORRECT way to implement the class `Quarantine` based on the class diagram? Assume that the class `Date` and `SelfQuarantine` have been implemented correctly.

-
  ```
  class Quarantine {
      string place;
      int duration;
      Date date;
      SelfQuarantine *selfqrtine;
  };
  ```

-
  ```
  class Quarantine {
      string place;
      int duration;
      Date *date;
      SelfQuarantine selfqrtine;
  };
  ```

-
  ```
  class Quarantine : public Date {
      string place;
      int duration;
      SelfQuarantine selfqrtine;
  };
  ```

-
  ```
  class Quarantine : public Date {
      string place;
      int duration;
      SelfQuarantine *selfqrtine;
  };
  ```

```
 1  // Program
 2
 3  class Parent
 4  { public:
 5      void print()
 6      {
 7          int y=15;
 8             cout<<"In Parent : ";
 9          cout<<y<<endl;
10      }
11  };
12
13  class Child: public Parent
14  {
15  public:
16      int x;
17      void print()
18      {
19          x = 15;
20          cout<<"In Child : ";
21          cout<< x <<endl;
22      }
23      Child(){x = 10;}
24  };
25
26  int main(void)
27  {
28      Parent *p, pr;
29      Child c;
30      p = &c;
31      __________
32      __________
33      __________
34      return 0;
35  }
36
```

56. Given the program above. Determine the correct codes for Line 31, Line 32 and Line 33 that can display the output as below :

```
In Parent : 15
In Parent : 15
In Child : 15
```

| | |
|---|---|
| `p->print();`<br>`pr.print();`<br>`c.print();` | `c.print();`<br>`pr.print();`<br>`c.print();` |
| `p->print();`<br>`pr->print();`<br>`c->print();` | `p.print();`<br>`pr.print();`<br>`c->print();` |

```
// Program

class Parent
{
public:

      Parent( )
      {
       cout<<"B ";
      }

      ~Parent()
      {
       cout<<"L " ;
      }

};

class Child : public Parent
{
public:
      Child( )
      {
       cout<<"U ";
      }

      ~Child()
      {
       cout<<"R " ;
      }

};
```

57. Consider the program as above. What would the following code segments print?

```
Parent p;
Child  child;
```

- B B U R L L
- B U B R L L
- B L B L U R
- B B L L U R

```
class House
{
   public : int no;
     void info()
     {
          cout<<"Base class ";
     }
};

class HomeStay:public House
{
    public :
     void info()
     {
               cout<<"Derived class ";
     }
};

int main() {
   House h;
   HomeStay hs;
   hs.info();
   h.info();

   return 0;
}
```

58. What is the output of the program above?

- `Derived class Base class`
- `Base class Derived class`
- `Base class Base class`
- `Derived class Derive class Base class`

```
#include <iostream>
using namespace std;

int main()
{
        int v [] = {5, 2, 5, 2, 2};
        int j=0;

        for (int j=0; j<5; j++){
                for (int i=0; i<v[j]; i++){

                        char a= '+';
                        cout<<a;
                }
                cout<<endl;
        }

        return 0;
}
```

59. Which one of the following programs that uses container to replace array in the program above is executable with the same output?

●
```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
      vector<int> v = {5, 2, 5, 2, 2};
      for (auto i : v){
             string output(i, '+');
             cout << output << "\n";
      }
      return 0;
}
```

●
```
#include <iostream>
#include <map>
using namespace std;

int main()
{
      map<int> v = {5, 2, 5, 2, 2};
      for (auto i : v){
             string output(i, '+');
             cout << output << "\n";
      }
      return 0;
}
```

●
```
#include <iostream>
#include <set>
using namespace std;

int main()
{
      set<int> v = {5, 2, 5, 2, 2};
      for (auto i : v){
             string output(i, '+');
             cout << output << "\n";
      }
      return 0;
}
```

●
```
#include <iostream>
#include <multiset>
using namespace std;

int main()
{
      multiset<int> v = {5, 2, 5, 2, 2};
      for (auto i : v){
             string output(i, '+');
             cout << output << "\n";
      }
      return 0;
}
```

```
class Rectangle
{
    private:
        double width;
        double length;
    public:
        void setData(double w, double l){ width = w; length = l;}
        double getWidth(){ return width; }
        double getLength(){ return length; }
        double getArea(){ return width * length; }
};
```

60. Based on the class declaration above, which one of the Rectangle class declaration is CORRECT after rewrite as a template that will accept any data type for these members?

●
```
template <class T>
class Rectangle
{
    private:
        T width;
        T length;
        T area;
    public:
        void setData(T w, T l){ width = w; length = l;}
        void calcArea(){ area = width * length; }
        T getWidth(){ return width; }
        T getLength(){ return length; }
        T getArea(){ return area; }
};
```

●
```
template <class T>
class Rectangle
{
    private:
        width;
        length;
        area;
    public:
        void setData(T w, T l){ width = w; length = l;}
        void calcArea(){ area = width * length; }
        T getWidth(){ return width; }
        T getLength(){ return length; }
        T getArea(){ return area; }
};
```

●
```
template <class T>
class Rectangle
{
    private:
        T width;
        T length;
        T area;
    public:
        void setData(T w, T l){ width = w; length = l;}
        void calcArea(){ area = width * length; }
        getWidth(){ return width; }
        getLength(){ return length; }
```

|   |   |
|---|---|
|   | `        getArea(){ return area; }`<br>`};` |
| ● | `template <class T>`<br>`class Rectangle`<br>`{`<br>`    private:`<br>`        T width;`<br>`        T length;`<br>`        T area;`<br>`    public:`<br>`        void setData(double w, double l){ width = w; length =`<br>`        l;}`<br>`        void calcArea(){ area = width * length; }`<br>`        T getWidth(){ return width; }`<br>`        T getLength(){ return length; }`<br>`        T getArea(){ return area; }`<br>`};` |

60.

```
#include<iostream>
#include<string>
using namespace std;

int main() {

    const string VIRUS = "Covid-19";
    string data = "corona";

    try{
        if (data != "free virus") throw VIRUS;
        cout << data << endl;
    }
    catch (string e){
        cout << "Exception occured: Thrown value is " << e << endl;
    }

   return 0;
}
```

Exception occurred: Thrown value is Covid-19

Exception occurred: Thrown value is VIRUS

Exception occurred: Thrown value is corona

corona