Question 1 ................................................................. (6 marks)

1.1 The boolean expression \( a \land (b \lor c) \land d \) evaluates to:
   A. Always false if \( a \) is false.  B. Always true if \( a \) or \( d \) are true.  C. True or false, only depended on the values of \( b \) and \( c \).  D. Always false if \( c \) is true.

1.2 References can be reinitialized in C++.
   A. For all data types and objects.  B. Never.  C. Only for primitive data types.  D. Only if it wasn’t initialized before.

1.3 In C++, \texttt{class SuperMan : private NormalMan\{\}} is an example of:

1.4 The following is true with regards to C++ inheritance:
   A. Inheritance can be public, protected or private.  B. The maximum number of subclasses that can inherit from a base class is limited to 64.  C. The maximum depth of inheritance is limited to 32 levels.  D. All of the above.

1.5 Design patterns were specifically designed for C++, because
   A. Design patterns were not specifically designed for C++.  B. Availability of pointers which are not present in other languages.  C. UML, used to model patterns, was specifically created for C++.  D. C++ was one of only few languages when design patterns were first published.

1.6 Design patterns are
   A. The only way of solving a programming problem.  B. A general reusable solution to a commonly programming problem.  C. Useless in most programming problems.  D. None of the above.

Question 2 ................................................................. (8 marks)

Indicate the most appropriate answer for the give UML diagram.

2.1 The diagram above is a

2.2 Element W in the diagram is a
   A. State node.  B. Transition node.  C. Activity node.  D. Class node.
2.3 Element X in the diagram is a

2.4 Element Y in the diagram is a

2.5 Element Z in the diagram between the [ ] brackets is a

**Question 3** .................................................................(7 marks)

Consider the following class diagram and complete the statements which follow.

![Class Diagram](image)

3.1 ____?____ is an attribute.
   A. name  B. Chatroom  C. ModuleChatroom()  D. cr

3.2 ____?____ is a class name.
   A. Chatroom  B. name  C. ModuleChatroom()  D. cr

3.3 ____?____ is a parameter.
   A. Chatroom  B. cr  C. name  D. ModuleChatroom()

3.4 ____?____ is a constructor.
   A. cr  B. ModuleChatroom()  C. Chatroom  D. name

3.5 The # sign is used in UML class diagrams to indicate:
   A. Protected members.  B. Virtual functions.  C. Constructors.  D. Not part of UML class diagrams.

3.6 A UML class diagram relationship with a hollow diamond at the end indicates:

The object diagram below represents the state of instances of the classes in the class diagram at a moment in time.

![Object Diagram](image)

3.7 ____?____ is a class.
   A. sarah  B. ModuleChatroom  C. name  D. None of the options

3.8 ____?____ is an object.
   A. ModuleChatroom  B. sarah  C. name  D. None of the options

3.9 ____?____ is a method defined in the class.
   A. sarah  B. ModuleChatroom  C. name  D. None of the options

3.10 In UML object diagrams, an object with the title :Student is:
   A. A class.  B. An uninitialized object.  C. A state transition.  D. An unnamed instance.
Question 4

Match the intent of the pattern to the pattern name. (7 marks)

4.1 Define an interface for creating an object, but let subclasses decide which class to instantiate. This pattern lets a class defer instantiation to subclasses.

4.2 Without violating encapsulation, capture and externalise an object’s internal state so that the object can be restored to this state later.

4.3 Cloning a large army of Storm Troopers.
   A. State.  B. Template Method.  C. Prototype.  D. a or c.

4.4 Creating peanut, cherry, and chocolate candy for the CandyTycoon video game.
   A. State.  B. Strategy.  C. Template Method.  D. None of the above.

4.5 During the sorting of a numbered array, let subclasses execute steps or the entire sorting algorithm.
   A. Strategy.  B. Template Method.  C. a and b.  D. None of the above.

4.6 The following is not a design pattern:

4.7 The following is not a creational design pattern:

Question 5

For each of the patterns given below, identify which participant does not belong to the pattern. (4 marks)

5.1 Abstract Factory
   A. ConcreteFactory  B. AbstractProduct  C. ConcreteProduct  D. Creator

5.2 Strategy
   A. Context  B. Strategy  C. CurrentState  D. ConcreteStrategy

5.3 Prototype
   A. Prototype  B. ConcretePrototype  C. Client  D. ConcreteClone

5.4 Template Method
   A. AbstractClass  B. ConcreteClass  C. TemplateClass  D. Client

Question 6

Each of the following questions describes a scenario which can be implemented with using design patterns. For each question you have to identify the most suited design pattern and participants from the options given to you. (14 marks)

6.1 A famous actress needs a number of different outfits, each suited to a particular occasion:
   - Casual wear for activities in public. A personal stylist who follows fashion trends closely is responsible for this type of outfit.
   - A costume to wear while shooting movies. The costume department on a movie set is responsible for producing the correct costume(s).
   - Formal evening dresses to wear to awards ceremonies and movie premiers. A fashion designer takes responsibility for producing an evening dress.

Assume you are tasked with producing an application which simulates an actress obtaining different types of outfits.

a) Which design pattern is most appropriate for this scenario?

b) Which role would a fashion designer assume?
   A. ConcreteFactory.  B. ConcreteClass.  C. ConcreteCreator.  D. PrototypeManager.
c) Which role would an outfit assume?

d) Which role would an evening dress assume?

6.2 Various sorting algorithms have steps in common. Among the differences between the algorithms are where they start and stop comparing elements in an array as well as whether or not they will swap two elements under scrutiny to provide an array sorted in ascending or descending order, dependant on a `compare` operation. A particular algorithm will know how to implement its differences from other algorithms.

   a) Which design pattern is most appropriate for this scenario?

   b) Which role would a specific sorting algorithm assume?
      A. Originator.  B. ConcretePrototype.  C. ConcreteClass.  D. Leaf.

   c) Which role would a super class for all sorting algorithm assume?

6.3 You and your colleagues were tasked with creating a database application. Each member of the team was assigned a different operation on the database to implement. Your task is to implement the redo operation for a database object which may be performed multiple times.

   a) Which design pattern is most appropriate for this scenario?

   b) Which role would a database object assume?

6.4 A new strategy game is currently under development. The developer has created two classes representing units, namely `Ornithopter` and `Devastator`. Both of these classes inherit from an abstract class `Unit`. Having this arrangement will not only make it easy to handle units uniformly throughout the code, but will also allow for the easy addition of more units, like for example, if an expansion pack is created for the game at a later date. There is also a class named `ConstructionYard` which is able to create units of any type upon request. This class encapsulates two objects, one of type `Ornithopter` and one of type `Devastator`.

   a) Which design pattern is most appropriate for this scenario?

   b) Which role would the `Unit` class assume?

   c) Which role would the `ConstructionYard` class assume?
      A. ConcreteCreator.  B. Creator.  C. PrototypeManager.  D. ConcreteFactory.

   d) Which role would an `Ornithopter` assume?
      A. ConcreteProduct.  B. Product.  C. ConcreteProductA2.  D. ConcretePrototype.

   e) Which role would the game play in the pattern?