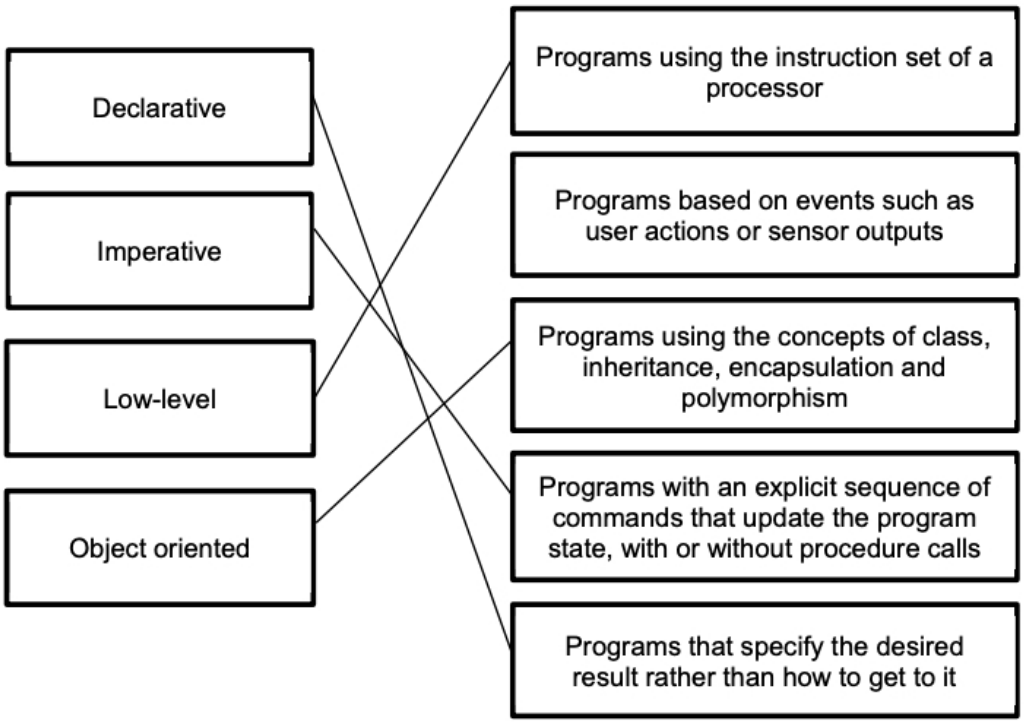
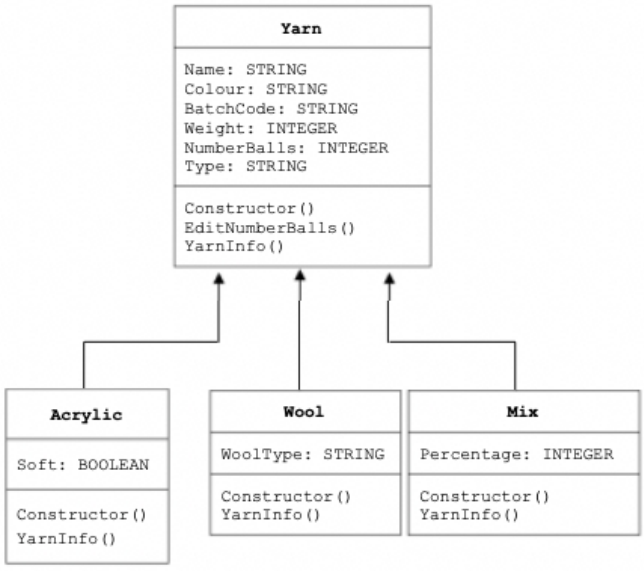
Question	Answer	Marks
9(a)	**One** mark for each correct marking point **(Max 2)** • Imperative languages use variables • … which are changed using (assignment) statements • … they rely on a method of repetition / iteration. • The statements provide a sequence of commands for the computer to perform • … in the order written / given • … each line of code changes something in the program run.	**2**
9(b)	**One** mark for each correct marking point **(Max 2)** • Instructs a program on what needs to be done instead of how to do it • ... using facts and rules • … using queries to satisfy goals. • Can be logical or functional • Logical - states a program as a set of logical relations • Functional – constructed by applying functions to arguments / uses a mathematical style	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>9(c)</td><td>One mark for each correct programming paradigm (Max 4)
<table>
<tr><th>Program code example</th><th>Programming paradigm</th></tr>
<tr><td><pre>male(john).
female(ethel).
parent(john, ethel).</pre></td><td>Declarative</td></tr>
<tr><td><pre>FOR Counter = 1 TO 20
    X = X * Counter
NEXT Counter</pre></td><td>Procedural / imperative</td></tr>
<tr><td><pre>Start: LDD Counter
       INC ACC
       STO Counter</pre></td><td>Low-level / assembly</td></tr>
<tr><td><pre>public class Vehicle
{
    private speed;
    public Vehicle()
    {
        speed = 0;
    }
}</pre></td><td>Object oriented / (OOP)</td></tr>
</table></td><td>4</td></tr>
</table>

Question	Answer	Marks
2	**One** mark for each **single** correct line from **Programming Paradigm** to **Description** **Programming Paradigm** / **Description**  	**4**

Question	Answer	Marks
2(a)	`type(caracal, wild).` `hair(caracal, short).`	2
2(b)	`persian`	1
2(c)(i)	`type(Pet, domestic).`	1
2(c)(ii)	`spots(WildSpotty, yes)` `,type(WildSpotty, wild).`	2

Question	Answer	Marks

7(a)	**1 mark** per point
	`Acrylic` has attribute `Soft` of type `BOOLEAN`
	`Wool` has attribute `WoolType` with suitable data type
	`Acrylic` **and** `Wool` have method `YarnInfo()`
	`Acrylic`, `Wool` and `Mix` **at least one** inherit (one arrow correct) from `Yarn` ...
	... `Acrylic`, `Wool` and `Mix` **all** inherit (all arrows correct) from `Yarn`
	 
		5

Question	Answer	Marks
7(b)	Properties **max 2**: • the data items / attributes • the data types // characteristics • defined in a class Methods **max 2**: • the procedures/ functions / programmed instructions in a class / super class / base class • … implementing the behaviours • … that act on the properties / attributes Inheritance **max 2**: • Methods and properties / attributes contained in one class/ super class / base class • Are made available to / reused by another class/ derived class	**6**

Question	Answer	Marks
11(a)	**One** mark for each correct OOP term definition: • **Instance** – an occurrence of an object // a specific object based on the class // an instantiation of a class. • **Inheritance** – the capability of defining a new class of objects that has all the attributes and methods from a parent class. • **Polymorphism** – allows the same method to take on different behaviours depending on which class is instantiated // methods can be redefined for derived classes.	**3**

Question	Answer	Marks

11(b)	**One** mark for each point:
	
	• `Car` and `ENDCLASS`
	• Four declarations – must use the identifiers used in the assignments
	• Constructor header – must use `CarBodyType`
	• Two assignments – must use `CarMake`
	• Constructor identifier for the car model and the identifier in the Model assignment statement match
	
	`CLASS Car`
	`    PRIVATE Make : STRING`
	`    PRIVATE Model : STRING`
	`    PRIVATE BodyType : STRING`
	`    PRIVATE Fuel : STRING`
	`    PRIVATE NumberBuilt : INTEGER`
	`    PUBLIC PROCEDURE NEW (CarMake : STRING,`
	`    CarModel : STRING, CarBodyType : STRING)`
	`        Make ← CarMake`
	`        Model ← CarModel`
	`        BodyType ← CarBodyType`
	`        Fuel ← ""`
	`        NumberBuilt ← 0`
	`    ENDPROCEDURE`
	`    getFuel()`
	`    getNumberBuilt()`
	`ENDCLASS`
		5