

Programming Paradigm

Question No. 1

| 9(a) | <p>One mark for each correct marking point (Max 2)</p> <ul style="list-style-type: none">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) | <p>One mark for each correct marking point (Max 2)</p> <ul style="list-style-type: none">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 functionalLogical - states a program as a set of logical relationsFunctional – constructed by applying functions to arguments / uses a mathematical style | 2 | | | | | | | | | | |
| 9(c) | <p>One mark for each correct programming paradigm (Max 4)</p> <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> | Program code example | Programming paradigm | <pre>male(john) . female(ethel) . parent(john, ethel) .</pre> | Declarative | <pre>FOR Counter = 1 TO 20 X = X * Counter NEXT Counter</pre> | Procedural / imperative | <pre>Start: LDD Counter INC ACC STO Counter</pre> | Low-level / assembly | <pre>public class Vehicle { private speed; public Vehicle() { speed = 0; } }</pre> | Object oriented / (OOP) | 4 |
| Program code example | Programming paradigm | | | | | | | | | | | |
| <pre>male(john) . female(ethel) . parent(john, ethel) .</pre> | Declarative | | | | | | | | | | | |
| <pre>FOR Counter = 1 TO 20 X = X * Counter NEXT Counter</pre> | Procedural / imperative | | | | | | | | | | | |
| <pre>Start: LDD Counter INC ACC STO Counter</pre> | Low-level / assembly | | | | | | | | | | | |
| <pre>public class Vehicle { private speed; public Vehicle() { speed = 0; } }</pre> | Object oriented / (OOP) | | | | | | | | | | | |

Question No. 2

| 2 | <p>One mark for each single correct line from Programming Paradigm to Description</p> <table><tr><th>Programming Paradigm</th><th>Description</th></tr><tr><td>Declarative</td><td>Programs using the instruction set of a processor</td></tr><tr><td>Imperative</td><td>Programs based on events such as user actions or sensor outputs</td></tr><tr><td>Low-level</td><td>Programs using the concepts of class, inheritance, encapsulation and polymorphism</td></tr><tr><td>Object oriented</td><td>Programs with an explicit sequence of commands that update the program state, with or without procedure calls</td></tr><tr><td></td><td>Programs that specify the desired result rather than how to get to it</td></tr></table> | Programming Paradigm | Description | Declarative | Programs using the instruction set of a processor | Imperative | Programs based on events such as user actions or sensor outputs | Low-level | Programs using the concepts of class, inheritance, encapsulation and polymorphism | Object oriented | Programs with an explicit sequence of commands that update the program state, with or without procedure calls | | Programs that specify the desired result rather than how to get to it | 4 |
|----------------------|---|----------------------|-------------|-------------|---|------------|---|-----------|---|-----------------|---|--|---|---|
| Programming Paradigm | Description | | | | | | | | | | | | | |
| Declarative | Programs using the instruction set of a processor | | | | | | | | | | | | | |
| Imperative | Programs based on events such as user actions or sensor outputs | | | | | | | | | | | | | |
| Low-level | Programs using the concepts of class, inheritance, encapsulation and polymorphism | | | | | | | | | | | | | |
| Object oriented | Programs with an explicit sequence of commands that update the program state, with or without procedure calls | | | | | | | | | | | | | |
| | Programs that specify the desired result rather than how to get to it | | | | | | | | | | | | | |