1.

a) First generation languages (Machine languages): Languages which are at a very low level of abstraction, consisting of only binary numbers. Ex: 0101101110

Second generation languages (Assembly languages): Languages which gave us the option to give variables and operations somewhat understandable names. Ex: ADD #4

Third generation languages (High-level languages): Languages which are independent of the processor and resemble human languages. They are portable, meaning code developed on one system could be transferred and executed on another. Ex: FORTRAN, C++, Java

Fourth generation languages (Very high-level languages): Languages which are improved based on 3GLs, designed and developed to reduce the time, cost and effort needed to develop different types of software applications. Ex: Ruby, SQL, MatLab

Fifth generation languages (Artificial intelligence languages): Languages which mainly focus on constraint programming. These languages are mostly used in AI and ML research, and will “solve the problem for you”. Ex: Mercury, OPS5, Prolog

b)

|  |  |  |
| --- | --- | --- |
| **Imperative** | **Object-oriented** | **Declarative** |
| Develop a sequence of imperative commands that  manipulates the data to obtain the desired result | Programs are objects and controlled through predefined methods | Programs are description of the problems to be solved |
| The structure of the program can be clarified by dividing it to subprograms called  procedures. Procedures can be executed sequentially or concurrently | Foundation of most OOP languages is imperative | Declarative algorithms are non deterministic, so the execution order of phases of  the algorithm is not typically known. |
| Examples: C, Ada, Pascal, FORTRAN, Basic, Cobol, … | Examples: C++, Java, Python | Examples: Prolog , SQL |

2. Several languages belong exclusively to one paradigm; however, they might have functions in other paradigms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Languages** | **Imperative** | **Functional** | **Object-oriented** | **Declarative** |
| C | X |  | X |  |
| Java | X | X | X |  |
| Matlab | X | X | X |  |
| C++ | X | X | X |  |
| Prolog |  |  |  | X |
| Python | X | X | X |  |
| Rust | X | X | X |  |
| Lisp | X | X | X |  |
| R | X | X | X |  |
| Perl | X | X | X |  |
| SQL |  |  |  | X |
| JavaScript | X | X | X |  |
| PHP | X |  | X |  |
| C# | X |  | X |  |

3.

a) If all of the objective functions and restrictions in a mathematical programming model are represented by linear equations, the model is called a linear model. A nonlinear model is one in which one or more of the objective functions or constraints are represented by a nonlinear equation.

b) A dynamic model accounts for changes in the state of the system over time, whereas a static (or steady-state) model calculates the system in equilibrium and is thus time-invariant. Differential equations or difference equations are commonly used to represent dynamic models.

c) A deterministic model is one in which each set of variable states is determined uniquely by model parameters and sets of prior values of these variables; as a result, a deterministic model always performs the same way for a given set of initial conditions. In a stochastic model, randomness is present, and variable states are characterized by probability distributions rather than unique values.

4.

input count

import RunningFree

while true loop

    print "Loop " + count

    play(RunningFree.intro)

    play(RunningFree.verse1)

    play(RunningFree.chorus)

    play(RunningFree.verse2)

    play(RunningFree.chorus)

    play(RunningFree.bridge)

    play(RunningFree.chorus)

    play(RunningFree.verse3)

    play(RunningFree.chorus)

    play(RunningFree.outro)