Question 1: Programming Paradigms and Languages

What is a programming language?
(2 points)

Cos'è un linguaggio di programmazione?
(2 punti)

A programming language is a linguistic tool by means of which it is possible to formalize an algorithm in such a way that it is executable by a computer as a programmable machine.

Un linguaggio di programmazione è uno strumento linguistico attraverso il quale è possibile formalizzare un algoritmo in modo tale che esso sia eseguibile da un computer in quanto macchina programmabile.

Question 2: Discrete Mathematics Background

What does it mean to assume as primitive the concepts of element, set, and membership and how is the last one denoted?
(2 points)

Cosa significa assumere come primitivi i concetti di elemento, insieme e appartenenza e come si denota quest'ultimo?
(2 punti)

It means that they are not expressible in terms of simpler concepts. We write $a \in A$ to indicate that a is an element belonging to set A.

Significa che essi non sono riconducibili ad altri concetti più semplici. Scriviamo $a \in A$ per indicare che a è un elemento appartenente all'insieme A.

Question 3: Lambda Calculus

Explain the difference between the extensional view and the intensional view of the concept of function.
(2 points)

Spiegare la differenza tra visione estensionale e visione intensionale del concetto di funzione.
(2 punti)

The extensional view of the concept of function focuses on the set of ordered pairs constituting the function, while the intensional view relies on the computational procedure through which one obtains the result $b = f(a)$ starting from the argument a.

La visione estensionale del concetto di funzione si focalizza sull'insieme di coppie ordinate che costituiscono la funzione, mentre la visione intensionale è incentrata sul procedimento computazionale tramite il quale si ottiene il risultato $b = f(a)$ partendo dall'argomento a.

Question 4: Haskell and ghci

How was the language Haskell created?
(2 points)

Come nacque il linguaggio Haskell?
(2 punti)

The language Haskell was designed towards the end of the 1980's by a European and American academic committee to share all the most useful and innovative aspects deriving from the proliferation of more or less purely functional languages during those decades.

Il linguaggio Haskell venne progettato verso la fine degli anni 1980 da un comitato accademico europeo e americano per mettere a fattor comune gli aspetti più utili e innovativi derivanti dalla proliferazione di linguaggi funzionali più o meno puri durante quei decenni.

Question 5: Propositional Logic

What is a proposition and how is it interpreted?
(2 points)

Cos'è una proposizione e come viene interpretata?
(2 punti)

A proposition is a statement related to a single fact of which we can establish the truth. It is formalized as a logical variable and is interpreted through a truth assignment, which is a set of propositions. If the proposition belongs to the truth assignment, then it is true, otherwise it is false.

Una proposizione è un'affermazione relativa a un singolo fatto di cui possiamo stabilire la verità. È formalizzata come una variabile logica e viene interpretata attraverso un assegnamento di verità, che è un insieme di proposizioni. Se la proposizione appartiene all'assegnamento di verità, allora è vera, altrimenti è falsa.

Question 6: Predicate Logic

What is a predicate and how is it interpreted?
(2 points)

Cos'è un predicato e come viene interpretato?
(2 punti)

A predicate is a logical function, hence each of its uses is syntactically formed by a predicate symbol applied to a finite number of terms, each composed of symbols of constant, variable, and function. Given a domain, a predicate symbol is interpreted as a relation over that domain, hence the application of a predicate symbol to its terms is true if and only if the tuple of domain values resulting from the interpretation of the various terms belongs to the relation.

Un predicato è una funzione logica, quindi ogni suo uso è sintatticamente formato da un simbolo di predicato applicato a un numero finito di termini, ciascuno composto da simboli di costante, variabile e funzione. Dato un dominio, un simbolo di predicato viene interpretato come una relazione su quel dominio, quindi l'applicazione di un simbolo di predicato ai suoi termini è vera se e soltanto se la tupla di valori del dominio derivanti dall'interpretazione dei vari termini appartiene alla relazione.

Question 7: Refutation of Logical Formulas

What is a refutation algorithm?
(2 points)

Cos'è un algoritmo di refutazione?
(2 punti)

A refutation algorithm is an algorithm that determines the validity of a formula by studying the unsatisfiability of the negation of that formula.

Un algoritmo di refutazione è un algoritmo che determina la validità di una formula studiando l'insoddisfacibilità della negazione di quella formula.

Question 8: Prolog and gprolog

How was the language Prolog created?
(2 points)

Come nacque il linguaggio Prolog?
(2 punti)

The language Prolog was developed in 1972 by Colmerauer and Roussel for natural language processing, on the basis of the procedural interpretation of Horn clauses given by Kowalski and the resolution method of Robinson.

Il linguaggio Prolog venne sviluppato nel 1972 da Colmerauer e Roussel per l'elaborazione del linguaggio naturale, sulla base dell'interpretazione procedurale delle clausole di Horn data da Kowalski e del metodo di risoluzione di Robinson.

Exercise 1: Haskell Programming

Write a Haskell program that computes the number of occurrences of a character in a string, in a purely declarative style.
(7 points)

Scrivere un programma Haskell che calcola il numero di occorrenze di un carattere in una stringa, in stile dichiarativo puro.
(7 punti)

```
num_occ :: Char -> String -> Int
num_occ _ []                  = 0
num_occ c (x : xs) | x == c    = 1 + num_occ c xs
                   | otherwise = num_occ c xs
```

Exercise 2: Prolog Programming

Write a Prolog program that computes the maximum of a list of negative numbers, in a purely declarative style.
(7 points)

Scrivere un programma Prolog che calcola il massimo di una lista di numeri negativi, in stile dichiarativo puro.
(7 punti)

```
max([], 0).
max([X], X).
max([X | L], Z) :- L \== [], max(L, Y), comp_max(X, Y, Z).

comp_max(X, Y, X) :- X >= Y.
comp_max(X, Y, Y) :- X < Y.
```