CE8-R3: LOGIC AND FUNCTIONAL PROGRAMMING

NOTE:

1. **Answer question 1 and any FOUR questions from 2 to 7.**
2. **Parts of the same question should be answered together and in the same sequence.**

Time: 3 Hours **Total Marks: 100**

1.

a) Convert $(\forall x)(\forall y)((\forall z)(a(x,y,z) \vee B(y))) \rightarrow (\forall x) C(x,z)$ prenex normal form.

b) Write a SML program for computation GCD.

c) Who exceptions are intercepted and handled in SML?

d) What is the following program doing? Is this program recursive and iterative? Justify.

```
ABC (N,F) ← ABC (0,N,1,F).
ABC (I,N,T,F) ← I<N,  I1 is I + 1, T1 is T*T1,  ABC(I1,N,T1,F).
ABC (N,N,F,F)
```

e) Discuss Cut in Prolog.

f) Define satisfiability, Validity and consistency in prepositional logic.

g) Define signature and functions in Functional programming, in brief.

(7x4)

2.

a) Define tree data type for a binary tree. Write a SML code to get depth of a binary tree.

b) Show that $((P \wedge \theta) \rightarrow R) \vee (\sim\theta \rightarrow \sim R)$ is valid using semantic tableux method.

c) Write iterative program in Prolog to add the elements (integers) of a given list.

(6+6+6)

3.

a) Write a Prolog code for Quick sort.

b) Prove $\{(p_i \rightarrow p_j), (\neg(p_j \rightarrow p_k) \rightarrow \neg p_i)\} \vdash (p_i \rightarrow p_k)$.

c) What is Resolution fefutation? Display a Resolution Refutation of

$\{\{\neg p_i, p_j\}, \{\neg p_j, p_k \neg p_i\}, \{p_i\}\{\neg p_k\}\}$

(6+4+4+4)

4. Let $\sum$ = {zero, succ, pred, plus} be a signature. Let I be the interpretation in Z, the set of integers such that I(zero)=0, I(succ)=$\lambda x[x+1]$, I(pred)=$\lambda x[x-1]$ and I(plus)=$\lambda(x, y)[x+y]$?

a) Define the set of normal forms (subset of T_Σ).

b) Define rewrite rules to compute the normal form of each term.

c) Define the Unique Σ -homomorophism $h_z : T_\Sigma \rightarrow Z$.

d) Let $I`$ be another interpretation in E, the set of all even inters with $I`$ (zero) = 0

$I`$ (succ) = $\lambda x[x+2]$, $I`$ (pred) = $\lambda x[x-2]$ and I (plus) = $\lambda(x,y)[x+y]$. Show that the two interpretations are isomorphic.

(4+4+4+6)

5.

a) Let b denotes any base type and let the following grammer define the language of simple types for the simply-typed Lambda Calculus.

$$\tau ::b|t|(\tau \to \tau)$$

Let Γ be a set of type assignments for variables. Then prove that substitution preserves types in the simply typed lambda calculus, i.e. x:α, M:α, and L:β then L{M/x}:β.

b) Find most general type assignments for the combinatory $S \overset{\Delta}{=} \lambda xyz[((xz)(yz))]$ in the second order language of types defined by

$$\tau ::b|t|(\tau \to \tau)$$
$$\tau ::\tau \,|\forall\, t[\pi]$$

where b denotes any base type and t denotes a type variable.

c) How control in Prolog is characterized with the help of an example. Show how response to a query is affected?

(6+6+6)

6.

a) Let derivatives (Y,X,Z) denote that the derivative of Y with respect to X is Z. Given the following facts:
derivative(N,X,0).
derivative(X,X,1).
derivative(sin(X),X,cos(X)).
derivative(cos(X),X,-sin(X)).
derivative(exp(X),X,exp(X)).
derivative(log(X),X,1/X).
write logic programming rules to compute derivatives of

i) Sum of two expressions.
ii) Difference of two expressions.
iii) Product of two expressions.
iv) Quotient of two expressions.

In particular make sure your definitions can calculate the derivatives of expressions such as 3*exp(5*X)*sin(2*X)+cos(X)/log(X).

b) Translate the following argument into first order logic and prove it using resolution.
Whoever visited the building was observed. Anyone who had observed Ajay, would have remembered him. Nobody remembered Ajay. Therefore Ajay did not visit the building.

(9+9)

7.

a) Let $S \overset{\Delta}{=} \lambda x\lambda y\lambda z[((x\ z)(y\ z))]$ and $K \overset{\Delta}{=} \lambda x\lambda y[x]$. Prove that application is not associative by showing that:

$((S\ K)\ K) \neq_{\beta} (S\ (K\ K))$

b) In all of mathematics, function composition denoted by the infix operator o is associative. That is, if f : A $\to$ B, g: B $\to$ C and h: C $\to$ D are any three functions, then the composition f o g : A $\to$ C is defined as the function (f o g)(a) = g(f(a)). It is then easy to see that o is associative, i.e.

(f o g) o h = f o (g o h)

Defined a lambda-expression called compose and prove that it is associative.

(9+9)