(3 Hours)

(Total Marks: 80)

N.B.: 1. Question No.1 is compulsory.
2. Answer any three out of remaining questions.
3. Assume suitable data if necessary.
4. Figures to the right indicate full marks.

Q1. a) Explain the difference between declarative and imperative programming paradigms. **(05)**

b) Explain lifecycle of a thread. **(05)**

c) Explain different types of Inheritance in OOP. **(05)**

d) Explain what is concept of higher order function? Explain working of any 2 higher order functions from Haskell Prelude library **(05)**

Q2. a) What are Scripting Languages? Explain characteristics of scripting languages. **(10)**

b) What is the role of an Exception Handler in a programming language? Briefly explain important tasks it performs. **(10)**

Q3. a) Explain different storage allocation mechanisms. **(10)**

b) What is logic programming? Explain Facts, Rules and Queries along with an example. **(10)**

Q4. a) Explain Type and Type classes in Haskell. **(10)**

b) Explain the different communication and synchronization techniques in Concurrent Programming model. **(10)**

Q5.

a) Define a Haskell function named "addUs" that adds 2 input numbers. Using this function as a building block, define a Haskell function "multiplyUs" that multiplies two input numbers. The multiplyUs function should cater to following:

i. Inputs may be signed numbers e.g. "multiplyUs (-2) * (3)" should result in "-6" and "multiplyUs (-2) * (-6)" should result in "12"

ii. It should use guard expressions and recursion.

iii. No need to write the main function to do user interaction. Writing definition for "addUs" and "multiplyUs" is sufficient**.** Explain type signature for your code.

b) Explain the unification and resolution in prolog with example. **(10)**

B7E456B27D5ED157DC53AF35215CFB26

Q6. Short note on: **(Any 4)** **(20)**

a) Static Scoping vs. Dynamic Scoping.
b) Types of bindings in Programming Languages.
c) Lazy versus Eager evaluation order for function parameters.
d) List operations in Prolog.
e) Need for Thread Synchronization in concurrent programming.

B7E456B27D5ED157DC53AF35215CFB26