Reg No.:_____ Name:_____

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**

Fourth Semester MCA Degree Regular and Supplementary Examination July 2021

# Course Code: RLMCA262
# Course Name: FUNCTIONAL PROGRAMMING

Max. Marks: 60 Duration: 3 Hours

## PART A

***Answer all questions, each carries 3 marks.*** Marks

1 Explain the concept of currying with the help of an example. (3)

2 State and explain the properties of bijective function with the help of an example. (3)

3 Which of the following are legal list constructions, also write the contents in list? (3)

a) list1 = 1 : [1,2]
b) list2 = [2] : [[1]]
c) list3 = [1] : [1]

4 Define Map and Filter using recursion and show their usage using examples. (3)

5 Explain the use of Tuples in functional programming with the help of an example. (3)

6 Explain and show how constructors are used to define an Enumerated data type with the help of an example. (3)

7 Define a Haskell function to remove even elements from a list. (3)

8 Write the lists generated by the following Haskell list expressions: (3)

a) [ x+1 | x <- [1..4]]
b) [x*2 | x <- [1..5], x*2 > 4]
c) [(i,j) | i <- [1,2], j <- [1..4],i<j ]

## PART B

***Answer any one question from each module. Each question carries 6 marks.***

### *Module I*

9 Explain any 3 data structures commonly used in functional programming. (6)

***OR***

10 a Let square x=x*x. Show different reduction sequences for square(square(3+2)) to normal form. (3)

| | | | |
|---|---|---|---|
| | b | Let pred x = x-1. Write and explain a definition to subtract two numbers using recursion. | (3) |

***Module II***

| | | | |
|---|---|---|---|
| 11 | | Explain with the help of examples, compare functional programming and imperative programming. | (6) |

***OR***

| | | | |
|---|---|---|---|
| 12 | | "In imperative languages, new values may be associated with the same name through command repetition.". Justify with suitable example. | (6) |

***Module III***

| | | | |
|---|---|---|---|
| 13 | | Explain the concept of functional composition with the help of an example. | (6) |

***OR***

| | | | |
|---|---|---|---|
| 14 | a | Explain lazy evaluation strategy with the help of suitable examples. | (3) |
| | b | Explain the usage of function guards and pattern matching with suitable examples. | (3) |

***Module IV***

| | | | |
|---|---|---|---|
| 15 | | Define, explain and state the use of any three list operations with suitable examples. | (6) |

***OR***

| | | | |
|---|---|---|---|
| 16 | | Prove x ^ (m + n) = (x ^ m) * (x ^ n) where ^ is the exponentiation operation using Mathematical Induction. Before proving, first define exponentiation using Recursion. | (6) |

***Module V***

| | | | |
|---|---|---|---|
| 17 | a | Explain how a type can be defined whose values depend on the values of other types along with examples. | (3) |
| | b | Explain how List data structure can be created as a Recursive data type along with examples. | (3) |

***OR***

| | | | |
|---|---|---|---|
| 18 | | Define binary search tree as a Recursive data type. Define and explain any two operations on it using that. | (6) |

***Module VI***

| | | | |
|---|---|---|---|
| 19 | a | Explain the use of zip function in functional programming with suitable example. | (3) |
| | b | Define a function in Haskell to find length of a list. | (3) |

***OR***

| | | | |
|---|---|---|---|
| 20 | | Define stack data structure in Haskell. Define and explain any two operations on it. | (6) |

*****