Set1

Roll No.: _____

Amrita Vishwa Vidyapeetham

Amrita School of Computing, Coimbatore

B.Tech Mid-Term Examinations – Feb 2025

Sixth Semester

Computer Science and Engineering

19CSE313 Principles of Programming Languages

Duration: Two hours | Maximum: 50 Marks

CO	Course Outcomes
CO01	Understand and write pure functional programs (especially in Haskell and Scala).
CO02	Understand and write concurrent programs in Java.
CO03	Formulate abstractions with higher order procedures.
CO04	Formulate abstractions with data.

Answer all questions

PART – A

1 In Haskell, nested lists can be used to represent matrices, though this approach is somewhat inefficient. For instance, the matrix $\begin{bmatrix} 3 & 2 \\ 3 & 1 \end{bmatrix}$ can be stored as a list where each row is represented as an element. For example, the matrix can be expressed as `[[3, 2], [3, 1]]`. A significant drawback of this representation is the potential for constructing an invalid matrix where rows have differing lengths.

Write a Haskell function `isVal` to verify whether all rows in a given matrix have the same size. Additionally, the function should print `TRIPLEMAT` if the matrix is 3x3 and `QUADMAT` if it is 4x4. Ensure the implementation is clean, robust, and passes all test cases. **[10] [CO01] [BTL3]**

```
ghci> isVal []
True
ghci> isVal [[1 ,2], [3 ,4], [5 ,6]]
True
ghci> isVal [[1 ,2 ,3], [4 ,5,6],[7,8,9]]
True;  TRIPLEMAT
```

2 Write a polymorphic function partitionList :: (a -> Bool) -> [a] -> ([a], [a]) that takes a predicate p and a list lst. It returns a pair of lists (match, noMatch) where:

- **match** contains elements of lst that satisfy the predicate p.

- **noMatch** contains elements of lst that do not satisfy p.

The relative order of elements in match and noMatch must be the same as their order in lst.

Ensure that your implementation passes the test cases provided below. [10] [CO03][BTL3]

i. partitionList even [1, 2, 3, 4, 5, 6]

Expected output: ([2,4,6],[1,3,5])

ii. partitionList (> 10) [5, 10, 15, 20]

Expected output: ([15,20],[5,10])

iii. partitionList (`elem` "aeiou") "haskell"

Expected output: ("ae","hsll")

PART – B

3 Briefly describe the working philosophies of both lazy and eager evaluations for the below Haskell code snippet:

[5] [CO01] [BTL2]

```
mySum :: [Int] -> Int
mySum [] = 0
mySum (x:xs) = x + mySum xs
-- Test case: A list that is large enough to show evaluation
differences
testList :: [Int]
testList = [1..1000000]

main :: IO ()
main = do
    print "Eager Evaluation:"
    print (mySum testList)
    print "Lazy Evaluation:"
    print (mySum testList)
```

4 Implement a Haskell function to update the i-th element of a list using recursion. Implement the same functionality using a non-recursive approach. **[2 + 3] [CO01] [BTL3]**

Input: L = [3,5,1,6,8,4]; i = 4; newVal = 15

Expected Output: L = [3,5,1,6,15,4]

5 Devise a curried function 'compose' that takes two functions, f and g, and returns their composition (i.e., a function that applies g and then f). Then, use a lambda function to create a composed function that: **[5] [CO03][BTL3]**

- Adds 5 to a number.
- Squares the result.
- Apply the composed function to the number 3.

Input : 3; Expected Output : 64

6 Define a Haskell function that takes one list of integers and two predicates and create a new list such that the predicate1 is applied on the odd numbers of the original list and predicate2 is applied on the even numbers of the list. **[5] [CO04] [BTL3]**

ghci> myFilter (>5) (<10) [1..20]
[2,4,6,7,8,9,11,13,15,17,19]

7 Write a Haskell function `removeVowels`, that removes all vowels from a list of strings. A sample execution is given below: **[5] [CO04] [BTL3]**

```
Input : removeVowels ["Hello", "World", "How", "Are", "You"]
Output : ["Hll","Wrld","Hw","r","Y"]
```

8 Fill in the missing terms of the below Haskell function `rotate`, which when supplied with a list `lst` and number of positions `N (+ve or -ve)`, rotates the list `N` places to the left. [5] [CO04] [BTL3]

```
ghci> rotate ['a','b','c','d','e','f','g','h'] 3 "defghabc"
ghci> rotate ['a','b','c','d','e','f','g','h'] (-2) "ghabcdef"

rotate :: [a] -> Int -> [a]
rotate xs n | n >= 0 = drop _____   _____ ++ take ____   _______
            | n < 0 = drop _____   _____ ++ take _____ _________
                    where len = _______ + length _______
```

****

Course Outcome /Bloom's Taxonomy Level (BTL) Mark Distribution Table

CO	Marks	BTL	Marks
CO01	20	BTL 1	0
CO02	-	BTL 2	5
CO03	15	BTL 3	45
CO04	15	BTL 4	-