AMRITA VISHWA VIDYAPEETHAM

23CSE212 – Principle of Functional Languages

BTech Computer Science and Engineering

Semester – 4

Lab Report

Academic Term: Jan – July 2025

Submitted by:

AM.SC.U4CSE23305

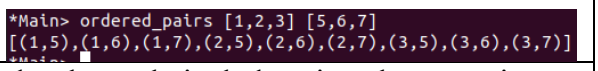
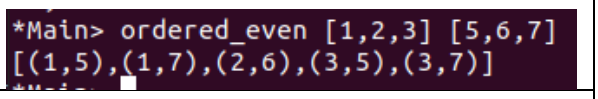
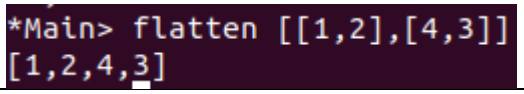
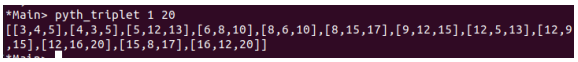
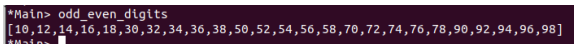
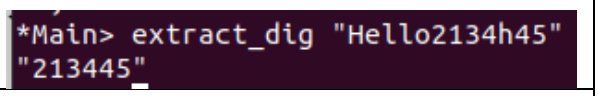
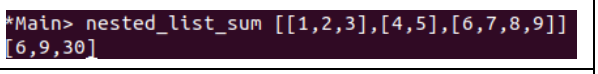
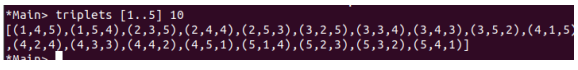
AMAN V SHAFEEQ

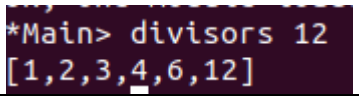
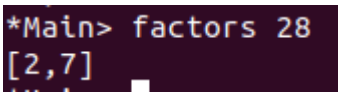
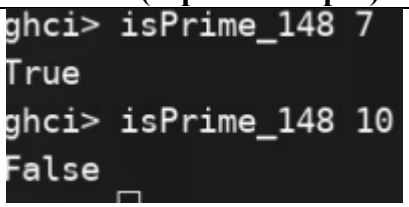
Lab Report

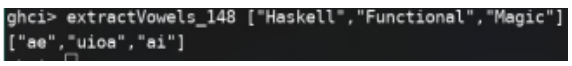
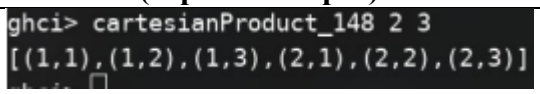
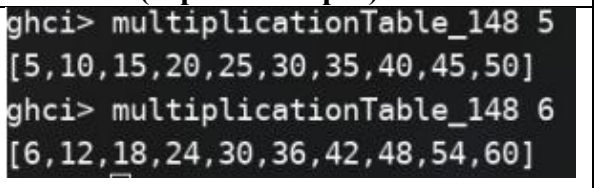
23CSE212 – Principles of Programming Languages

Criteria	Excellent	Good	Poor
Timely Submission			
Correctness of lab assignment			
Total Marks			
Signed By Lab Instructor			

Lab Session No: 5 **Date: 11/3/25**	
Question 1: Generate a list of squares of numbers from 1 to 5 **CO** 2	
Code	**Testcases (Input & Output)**
[x*x | x<-[1..5]]	[1,4,9,16,25]
Question 2: Genrate a list of even numbers from m to n	
Code	**Testcases (Input & Output)**
[x*x | x<-[1..5], mod x 2 == 0]	[4,16]
Question 3: Generate a list of first n odd numbers **CO2**	
Code	**Testcases (Input & Output)**
n_odd :: Int->[Int] n_odd n = take n [x | x<-[1..], mod x 2 /= 0]	*Main> n_odd 5 [1,3,5,7,9]
Question 4: Generate a list of numbers from m to n that are divisible by 3 **CO** 2	
Code	**Testcases (Input & Output)**
div_three :: Int -> Int -> [Int] div_three m n = [x | x<-[m..n], mod x 3 ==0]	*Main> div_three 1 10 [3,6,9]
Question 5: Generate all ordered pairs (x,y) from x if from xs and y is from ys lists **CO** 2	
Code	**Testcases (Input & Output)**

ordered_pairs :: [Int] -> [Int] -> [(Int,Int)] ordered_pairs xs ys = [(x,y) | x<-xs, y<-ys]	
Question 6: Generate all pairs (x,y) from xs and ys but only include pairs where x+y is even **CO** 2	
Code	**Testcases (Input & Output)**
ordered_even :: [Int] -> [Int] -> [(Int,Int)] ordered_even xs ys = [(x,y) | x<-xs, y<-ys, (x+y) `mod` 2 == 0]	
Question 7: Flatten a nested list **CO** 2	
Code	**Testcases (Input & Output)**
flatten :: [[Int]]->[Int] flatten xs = foldr (++) [] xs	
Question 8: Define a function duplicate which will duplicate the list 3 times and produce a new list **CO** 2	
Code	**Testcases (Input & Output)**
pyth_triplet :: Int->Int->[[Int]] pyth_triplet m n = [[x,y,z] | x<-[m..n], y<-[m..n], z<-[m..n], x^2 + y^2==z^2]	
Question 9 **CO** 2	
Code	**Testcases (Input & Output)**
odd_even_digits = [x | x<-[10..99], (x mod 10) mod 2 == 0 && (x div 10) mod 2 /= 0]	
Question 10 **CO** 2	
Code	**Testcases (Input & Output)**
extract_dig :: String->String extract_dig xs = [x | x<-xs, isDigit x == True]	
Question 11 **CO** 2	
Code	**Testcases (Input & Output)**
nested_list_sum :: [[Int]]->[Int] nested_list_sum xss = [sum xs | xs<-xss]	
Question 12 **CO** 2	
Code	**Testcases (Input & Output)**
triplets :: [Int]->Int->[(Int, Int, Int)] triplets xs s = [(x,y,z) | x<-xs, y<-xs, z<-xs, x+y+z == s]	
Question 13: **CO** 2	

Code	Testcases (Input & Output)
divisors :: Int->[Int] divisors n = [x | x<-[1..n], n `mod` x == 0]	
Question 14: **CO** 2	
Code	**Testcases (Input & Output)**
isPrime k = if k > 1 then null [x | x <- [2..k - 1], k `mod` x == 0] else False factors :: Int->[Int] factors n = [x | x<-[1..n], n `mod` x == 0 && isPrime x == True]	
Question 15: **CO** 2	
Code	**Testcases (Input & Output)**
IsPrime_148 k = if k > 1 then null [x | x <- [2..k - 1], k `mod` x == 0] else False	

Question 16: **CO** 2	
Code	**Testcases (Input & Output)**
extractVowels_148 :: [String] -> [String] extractVowels_148 words = [[c | c <- word, c `elem` "aeiouAEIOU"] | word <- words]	
Question 17: **CO** 2	
Code	**Testcases (Input & Output)**
cartesianProduct_148 :: Int -> Int -> [(Int, Int)] cartesianProduct_148 m n = [(x, y) | x <- [1..m], y <- [1..n]]	
Question 18: **CO** 2	
Code	**Testcases (Input & Output)**
multiplicationTable_148 :: Int -> [Int] multiplicationTable_148 n = [n * x | x <- [1..10]]	
Question 19: **CO** 2	
Code	**Testcases (Input & Output)**

triangularNumbers_148 :: Int -> [Int] triangularNumbers_148 n = [sum [1..x] | x <- [1..n]]	ghci> triangularNumbers_148 5 [1,3,6,10,15] ghci> triangularNumbers_148 10 [1,3,6,10,15,21,28,36,45,55]
Question 20: **CO** 2	
Code	**Testcases (Input & Output)**
commonElements_148 :: Eq a => [a] -> [a] -> [a] commonElements_148 xs ys = [x | x <- xs, x `elem` ys]	ghci> commonElements_148 [1,2,3,4,5] [3,4,5,6,7] [3,4,5] ghci> commonElements_148 [1,2,11,5] [3,11,7] [11]
Question 21: **CO** 2	
Code	**Testcases (Input & Output)**
sumPairs_148 :: [Int] -> [Int] -> [Int] sumPairs_148 xs ys = [x + y | (x, y) <- zip xs ys]	ghci> sumPairs_148 [1,2,3] [4,5,6] [5,7,9] ghci> sumPairs_148 [10,2,30] [4,50,6] [14,52,36]
Question 22: **CO** 2	
Code	**Testcases (Input & Output)**
multiplyPairs_148 :: [Int] -> [Int] -> [Int] multiplyPairs_148 xs ys = [x * y | (x, y) <- zip xs ys]	ghci> multiplyPairs_148 [1,2,3] [4,5,6] [4,10,18] ghci> multiplyPairs_148 [5,2,6] [10,7,9] [50,14,54]
Question 23: **CO** 2	
Code	**Testcases (Input & Output)**
pairConsecutive_148 :: [a] -> [(a, a)] pairConsecutive_148 xs = zip xs (tail xs)	ghci> pairConsecutive_148 [1,2,3,4,5] [(1,2),(2,3),(3,4),(4,5)] ghci> pairConsecutive_148 [3,4,5,6] [(3,4),(4,5),(5,6)]
Question 24: **CO** 2	
Code	**Testcases (Input & Output)**
differences_148 :: [Int] -> [Int] differences_148 xs = [abs (x - y) | (x, y) <- zip xs (tail xs)]	ghci> differences_148 [1,2,15,30,25] [1,13,15,5] ghci> differences_148 [1,2,135,30,25] [1,133,105,5]
Question 20: **CO** 2	
Code	**Testcases (Input & Output)**

reversePairs_148 :: [a] -> [(a, a)] reversePairs_148 xs = zip xs (reverse xs)