# Lab Report

23CSE212 – Principles of Functional Languages

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

**Lab Session No: 2**
**Date: 06/03/25**

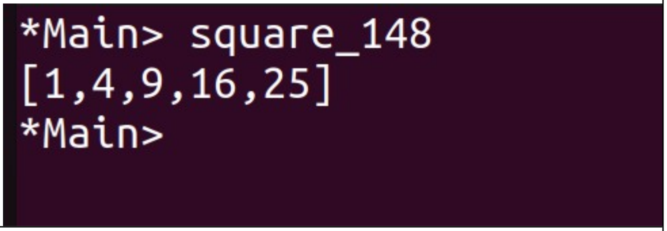
**CO2** : Develop Haskell programs to solve basic programming problems based on type classes, function definitions, higher-order functions, and list processing.

**Question 1**:

```
square_148 ::[Int]
square_148= [x^2 | x <- [1..5]]
```

**Input:**
`square_148`

**Output:**



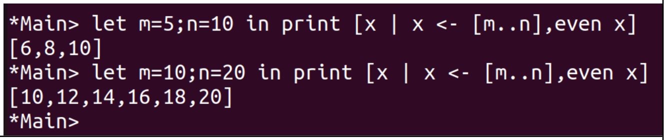
```
*Main> square_148
[1,4,9,16,25]
*Main>
```

**Question 2**:

```
let m=5;n=10 in print [x | x <- [m..n],even
x]
```

**Input:**
**5 10**
**10 20**

**Output:**



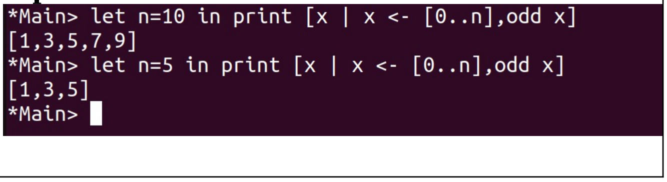
```
*Main> let m=5;n=10 in print [x | x <- [m..n],even x]
[6,8,10]
*Main> let m=10;n=20 in print [x | x <- [m..n],even x]
[10,12,14,16,18,20]
*Main>
```

**Question 3**:

```
let n=10 in print [x | x <- [0..n],odd x]
```

**Input:**
**10**
**5**

**Output:**



```
*Main> let n=10 in print [x | x <- [0..n],odd x]
[1,3,5,7,9]
*Main> let n=5 in print [x | x <- [0..n],odd x]
[1,3,5]
*Main>
```

# Lab Report

23CSE212 – Principles of Functional Languages

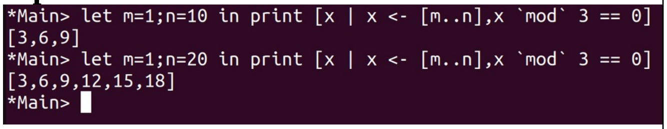
**Question 4**:

let m=1;n=10 in print [x | x <- [m..n],x `mod` 3 == 0]

**Input:**

**1 10**
**1 20**

**Output:**



```
*Main> let m=1;n=10 in print [x | x <- [m..n],x `mod` 3 == 0]
[3,6,9]
*Main> let m=1;n=20 in print [x | x <- [m..n],x `mod` 3 == 0]
[3,6,9,12,15,18]
*Main> 
```

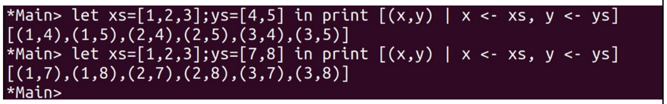
**Question 5**:

```
let xs=[1,2,3];ys=[4,5] in print [(x,y) | x
<- xs, y <- ys]
```

**Input:**

**[1,2,3] [4,5]**
**[1,2,3] [7,8]**

**Output:**



```
*Main> let xs=[1,2,3];ys=[4,5] in print [(x,y) | x <- xs, y <- ys]
[(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)]
*Main> let xs=[1,2,3];ys=[7,8] in print [(x,y) | x <- xs, y <- ys]
[(1,7),(1,8),(2,7),(2,8),(3,7),(3,8)]
*Main>
```

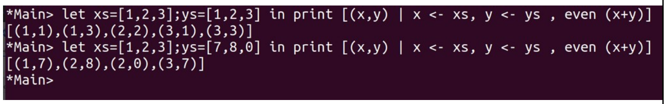
**Question 6**:

```
let xs=[1,2,3];ys=[1,2,3] in print [(x,y) | x
<- xs, y <- ys , even (x+y)]
```

**Input:**

**[1,2,3] [1,2,3]**
**[1,2,3] [7,8,0]**

**Output:**



```
*Main> let xs=[1,2,3];ys=[1,2,3] in print [(x,y) | x <- xs, y <- ys , even (x+y)]
[(1,1),(1,3),(2,2),(3,1),(3,3)]
*Main> let xs=[1,2,3];ys=[7,8,0] in print [(x,y) | x <- xs, y <- ys , even (x+y)]
[(1,7),(2,8),(2,0),(3,7)]
*Main>
```

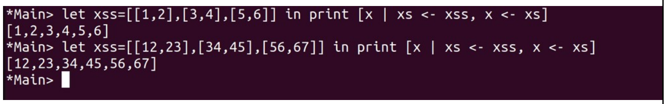
**Question 7**:

```
let xss=[[1,2],[3,4],[5,6]] in print [x | xs
<- xss, x <- xs]
```

**Input:**

**[[1,2],[3,4],[5,6]]**
**[[12,23],[34,45],[56,67]]**

**Output:**



```
*Main> let xss=[[1,2],[3,4],[5,6]] in print [x | xs <- xss, x <- xs]
[1,2,3,4,5,6]
*Main> let xss=[[12,23],[34,45],[56,67]] in print [x | xs <- xss, x <- xs]
[12,23,34,45,56,67]
*Main> 
```

# Lab Report

23CSE212 – Principles of Functional Languages

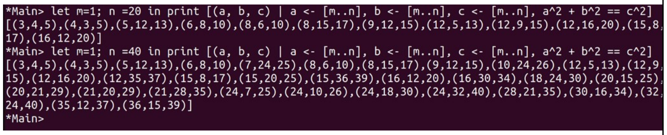
**Question 8**:

```
let m=1; n =20 in print [(a, b, c) | a <- [m..n], b <- [m..n], c <- [m..n], a^2 + b^2 == c^2]
```

**Input:**

**1 20**
**1 40**

**Output:**



```
*Main> let m=1; n =20 in print [(a, b, c) | a <- [m..n], b <- [m..n], c <- [m..n], a^2 + b^2 == c^2]
[(3,4,5),(4,3,5),(5,12,13),(6,8,10),(8,6,10),(8,15,17),(9,12,15),(12,5,13),(12,9,15),(12,16,20),(15,8,
17),(16,12,20)]
*Main> let m=1; n =40 in print [(a, b, c) | a <- [m..n], b <- [m..n], c <- [m..n], a^2 + b^2 == c^2]
[(3,4,5),(4,3,5),(5,12,13),(6,8,10),(7,24,25),(8,6,10),(8,15,17),(9,12,15),(10,24,26),(12,5,13),(12,9,
15),(12,16,20),(12,35,37),(15,8,17),(15,20,25),(15,36,39),(16,12,20),(16,30,34),(18,24,30),(20,15,25),
(20,21,29),(21,20,29),(21,28,35),(24,7,25),(24,10,26),(24,18,30),(24,32,40),(28,21,35),(30,16,34),(32,
24,40),(35,12,37),(36,15,39)]
*Main>
```

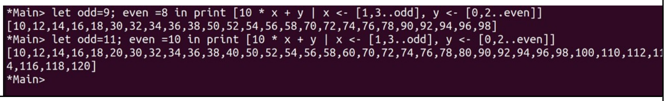
**Question 9**:

```
let odd=9; even =8 in print [10 * x + y | x <- [1,3..odd], y <- [0,2..even]]
```

**Input:**

**9 8**
**11 10**

**Output:**



```
*Main> let odd=9; even =8 in print [10 * x + y | x <- [1,3..odd], y <- [0,2..even]]
[10,12,14,16,18,30,32,34,36,38,50,52,54,56,58,70,72,74,76,78,90,92,94,96,98]
*Main> let odd=11; even =10 in print [10 * x + y | x <- [1,3..odd], y <- [0,2..even]]
[10,12,14,16,18,20,30,32,34,36,38,40,50,52,54,56,58,60,70,72,74,76,78,80,90,92,94,96,98,100,110,112,11
4,116,118,120]
*Main>
```

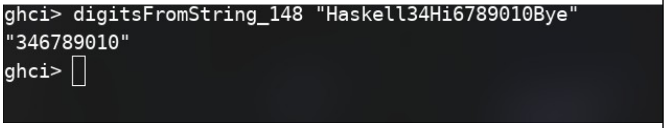
**Question 10**:

```
digitsFromString_148 :: String -> String
digitsFromString_148 str = [c | c <- str, c `elem` ['0'..'9']]
```

**Input:**

**"Haskell34Hi6789010Bye"**

**Output:**



```
ghci> digitsFromString_148 "Haskell34Hi6789010Bye"
"346789010"
ghci>
```

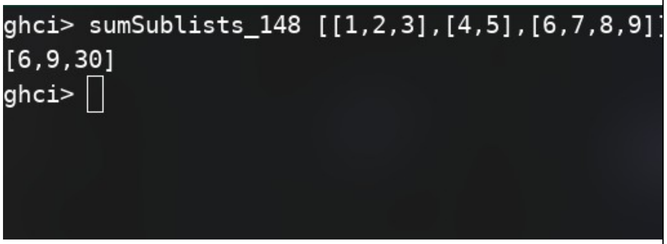
**Question 11**:

```
sumSublists_148 :: [[Int]] -> [Int]
sumSublists_148 xs = [sum x | x <- xs]
```

**Input:**

**[[1,2,3],[4,5],[6,7,8,9]]**

**Output:**



```
ghci> sumSublists_148 [[1,2,3],[4,5],[6,7,8,9]]
[6,9,30]
ghci>
```

# Lab Report

23CSE212 – Principles of Functional Languages

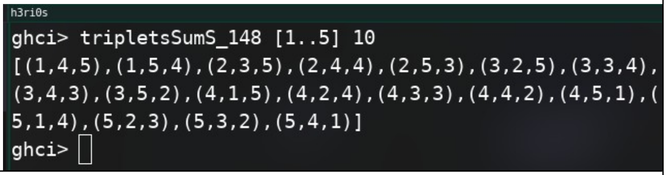
**Question 12**:

```
tripletsSumS_148 :: [Int] -> Int -> [(Int, Int, Int)]
tripletsSumS_148 xs s = [(x, y, z) | x <- xs, y <- xs, z <- xs, x + y + z == s]
```

**Input:**

**`[1..5] 10`**

**Output:**



```
ghci> tripletsSumS_148 [1..5] 10
[(1,4,5),(1,5,4),(2,3,5),(2,4,4),(2,5,3),(3,2,5),(3,3,4),
(3,4,3),(3,5,2),(4,1,5),(4,2,4),(4,3,3),(4,4,2),(4,5,1),(
5,1,4),(5,2,3),(5,3,2),(5,4,1)]
ghci>
```

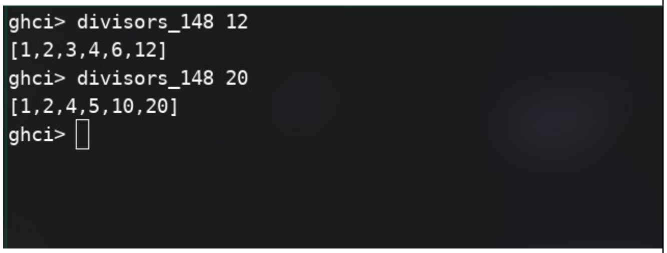
**Question 13**:

```
divisors_148 :: Int -> [Int]
divisors_148 n = [x | x <- [1..n], n `mod` x == 0]
```

**Input:**

**12**<br>
**20**

**Output:**



```
ghci> divisors_148 12
[1,2,3,4,6,12]
ghci> divisors_148 20
[1,2,4,5,10,20]
ghci>
```

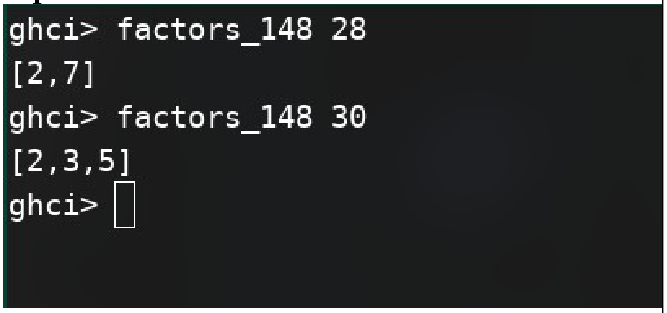
**Question 14**:

```
factors_148 :: Int -> [Int] factors_148 n = [x | x <- divisors_148 n, isPrime_148 x]
```

**Input:**

**28**<br>
**30**

**Output:**



```
ghci> factors_148 28
[2,7]
ghci> factors_148 30
[2,3,5]
ghci>
```

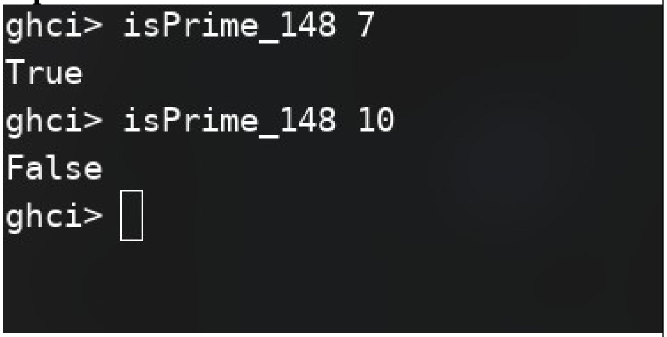
**Question 15**:

```
isPrime_148 :: Int -> Bool
isPrime_148 n = n > 1 && null [x | x <-
[2..n-1], n `mod` x == 0]
```

**Input:**

```
7
10
```

**Output:**



```
ghci> isPrime_148 7
True
ghci> isPrime_148 10
False
ghci> 
```

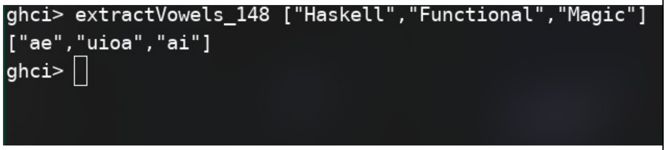
**Question 16**:

```
extractVowels_148 :: [String] -> [String]
extractVowels_148 words = [[c | c <- word, c
`elem` "aeiouAEIOU"] | word <- words]
```

**Input:**

```
["Haskell","Functional","Magic"]
```

**Output:**



```
ghci> extractVowels_148 ["Haskell","Functional","Magic"]
["ae","uioa","ai"]
ghci> 
```

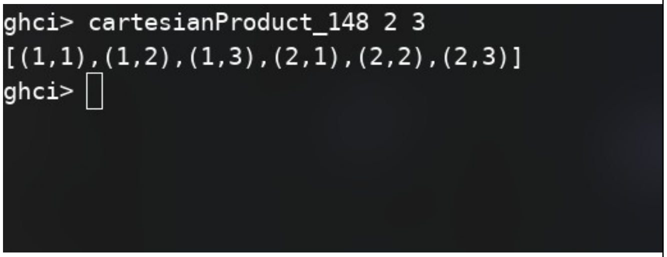
**Question 17**:

```
cartesianProduct_148 :: Int -> Int -> [(Int,
Int)]
cartesianProduct_148 m n = [(x, y) | x <-
[1..m], y <- [1..n]]
```

**Input:**

```
2 3
```

**Output:**



```
ghci> cartesianProduct_148 2 3
[(1,1),(1,2),(1,3),(2,1),(2,2),(2,3)]
ghci> 
```

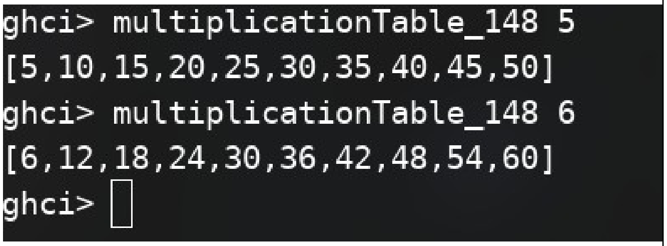
**Question 18**:

```
multiplicationTable_148 :: Int -> [Int]
multiplicationTable_148 n = [n * x | x <-
[1..10]]
```

**Input:**

```
5
6
```

**Output:**



```
ghci> multiplicationTable_148 5
[5,10,15,20,25,30,35,40,45,50]
ghci> multiplicationTable_148 6
[6,12,18,24,30,36,42,48,54,60]
ghci> 
```

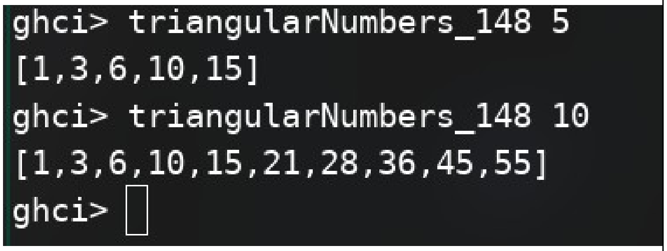
**Question 19**:

```
triangularNumbers_148 :: Int -> [Int]
triangularNumbers_148 n = [sum [1..x] | x <-
[1..n]]
```

**Input:**

```
5
10
```

**Output:**



```
ghci> triangularNumbers_148 5
[1,3,6,10,15]
ghci> triangularNumbers_148 10
[1,3,6,10,15,21,28,36,45,55]
ghci> 
```

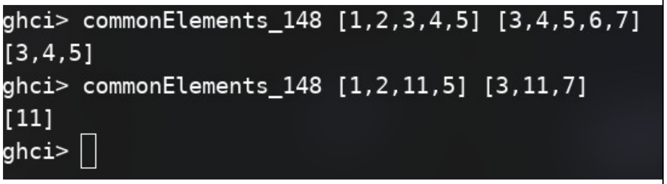
**Question 20**:

```
commonElements_148 :: Eq a => [a] -> [a] ->
[a]
commonElements_148 xs ys = [x | x <- xs, x
`elem` ys]
```

**Input:**

```
[1,2,3,4,5] [3,4,5,6,7]
[1,2,11,5] [3,11,7]
```

**Output:**



```
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]
ghci> 
```

# Lab Report

23CSE212 – Principles of Functional Languages

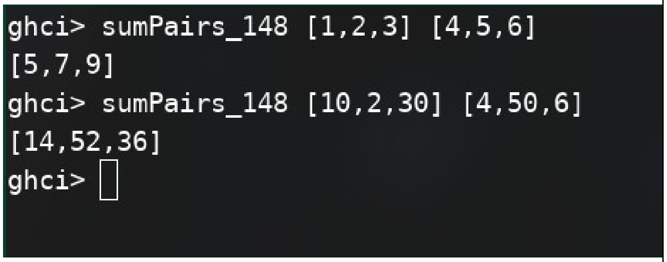
**Question 21**:

```
sumPairs_148 :: [Int] -> [Int] -> [Int]
sumPairs_148 xs ys = [x + y | (x, y) <- zip
xs ys]
```

**Input:**

**`[1,2,3] [4,5,6]`**
**`[10,2,30] [4,50,6]`**

**Output:**



```
ghci> sumPairs_148 [1,2,3] [4,5,6]
[5,7,9]
ghci> sumPairs_148 [10,2,30] [4,50,6]
[14,52,36]
ghci> 
```

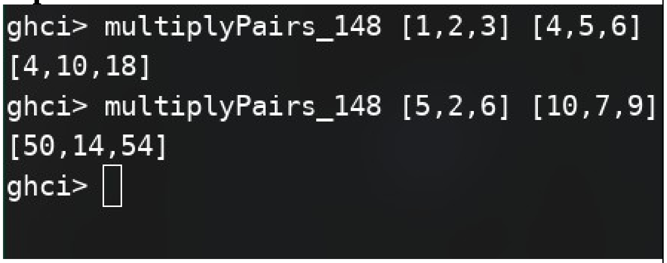
**Question 22**:

```
multiplyPairs_148 :: [Int] -> [Int] -> [Int]
multiplyPairs_148 xs ys = [x * y | (x, y) <-
zip xs ys]
```

**Input:**

**`[1,2,3] [4,5,6]`**
**`[5,2,6] [10,7,9]`**

**Output:**



```
ghci> multiplyPairs_148 [1,2,3] [4,5,6]
[4,10,18]
ghci> multiplyPairs_148 [5,2,6] [10,7,9]
[50,14,54]
ghci> 
```

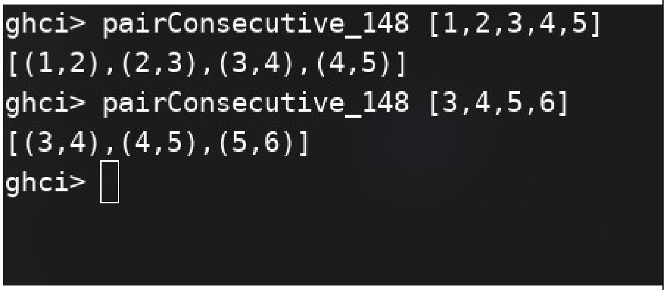
**Question 23**:

```
pairConsecutive_148 :: [a] -> [(a, a)]
pairConsecutive_148 xs = zip xs (tail xs)
```

**Input:**

**`[1,2,3,4,5]`**
**`[3,4,5,6]`**

**Output:**



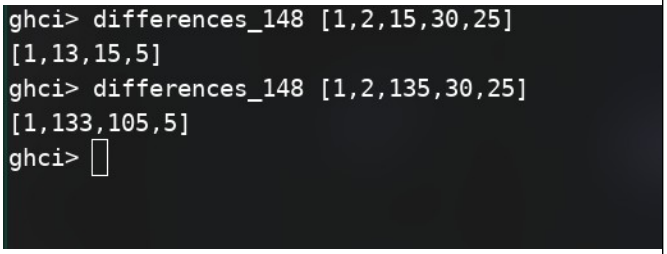
```
ghci> pairConsecutive_148 [1,2,3,4,5]
[(1,2),(2,3),(3,4),(4,5)]
ghci> pairConsecutive_148 [3,4,5,6]
[(3,4),(4,5),(5,6)]
ghci> 
```

**Question 24**:

```
differences_148 :: [Int] -> [Int]
differences_148 xs = [abs (x - y) | (x, y) <-
zip xs (tail xs)]
```

**Input:**
**[1,2,15,30,25]**
**[1,2,135,30,25]**

**Output:**



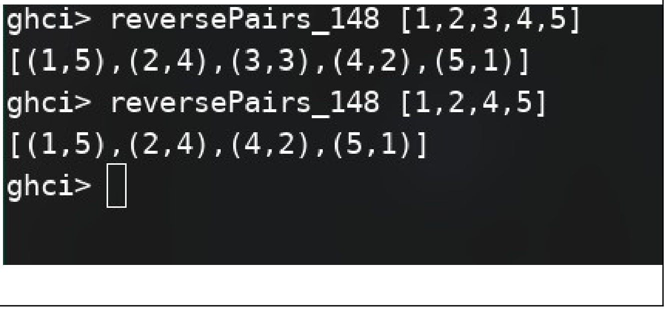
```
ghci> differences_148 [1,2,15,30,25]
[1,13,15,5]
ghci> differences_148 [1,2,135,30,25]
[1,133,105,5]
ghci> 
```

**Question 25**:

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

**Input:**
**[1,2,3,4,5]**
**[1,2,4,5]**

**Output:**



```
ghci> reversePairs_148 [1,2,3,4,5]
[(1,5),(2,4),(3,3),(4,2),(5,1)]
ghci> reversePairs_148 [1,2,4,5]
[(1,5),(2,4),(4,2),(5,1)]
ghci> 
```