# TUTORIAL 2

## LIST COMPREHENSION

**(All the Exercises should be done using List Comprehensions)**

1. Try doing the following List comprehensions and note down the change in the outputs.
   a. [(x,y) | x ← [1,2,3], y ← [4,5]]
      **[(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)]**
   b. [(x,y) | y ← [4,5], x ← [1,2,3]]
      **[(1,4),(2,4),(3,4),(1,5),(2,5),(3,5)]**
   c. [(x,y) | x ← [1..3], y ← [x..3]]
      **[(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)]**
2. Implement the following function and write down the output:
   pairs :: [a] → [(a,a)]
   pairs xs = zip xs (tail xs)
   **pairs [1,2,3,4]**
   **[(1,2),(2,3),(3,4)]**
3. Which of the following are legal list constructions?
   a. list1 = 1 : []
      **YES**
   b. list2 = 1 : [] : []
      **NO**
   c. list3 = 1 : [1]
      **YES**
   d. list4 = [] : [1]
      **NO**
   e. list5 = [1] : [1] : []
      **YES**
4. Using a predicate we can define a function that maps a positive integer to its list of factors as follows:
   factors :: Int → [Int]
   factors n = [x | x ← [1..n], n `mod` x == 0]
   a. By making use of this function check whether a number is prime or not
      **prime n = factorial n==[1,n]**
   b. Generate Prime numbers up to a limit
      **genprime n = [i | i <- [1..n], prime i==True ]**
5. Generate all Perfect numbers up to a limit n by making use of the above factors function
   **perfectno n= sum (factorial n)==n**
6. Write a function length' to get length of a list.(can use the built in function sum)
   **listlen []=0**
   **listlen (_:xs)=1 + listlen xs**

7. Write a function that takes a string and removes everything except uppercase letters from it.
   **import Data.Char**
   **s= “Hello World”**
   **p=filter isUpper s**
   **>p**
   **Output : “HW”**

8. Write a function to generate all triangles with sides equal to or smaller than 10.
   **[(a,b,c) | c<-[1..10],a<-[1..10],b<-[1..10],a+b>c,a+c>b,b+c>a]**
9. Implement the following function and note down the output
   count :: Char → String → Int
   count x xs = length [x1 | x1 ← xs, x == x1]
   **count 10 [10,2,10]**
   **O/p: 2 (Counts frequency of ‘n’ in list)**
10. Implement a function Pythagorean to check whether a given list is a Pythagorean triple. A triple (x,y,z) of positive integers is called Pythagorean if $x^2 + y^2 = z^2$.
    **pyth (a,b,c)**
    **| a^2+b^2 == c^2 = "YES,It is a pythogorian triple"**
    **| otherwise = "Not a pythogorian triple"**

11. Using a list comprehension, define a function
    pyths :: Int → [(Int,Int,Int)]

    that maps an integer n to all such triples with components in [1..n].

    For example:

    pyths 5

    [(3,4,5),(4,3,5)]

    **pyths n =[(a,b,c) | c<-[1..n],a<-[1..n],b<-[1..n], a^2 + b^2 == c^2]**