Q1. Evaluate the following expressions:

1. Tail [(True, False)] → (True, False)
2. Head [(“1”, “2”), (“3”, “4”)] → (“3”, “4”)
3. []:[[True], [False]] → [[], [True], [False]]
4. [“CS”, “653”] ++ [[]] → [“CS”,”653”,””]
5. (\ x y z -> z (y x)) [1, 2, 4, 8] tail head →

Q2. Function that takes a single digit number that returns + if the number is positive, - if negative

```
f :: (Num a, Ord a) => a -> Char
f x
   | x <= -10 || x >= 10 = error "Invalid input"
   | x > -10 && x < 0 = '-'
   | x == 0 = '0'
   | otherwise = '+'
```

Q3. Function that checks if a triplet is a pythagorean triplet

```
isTriplet :: (Num a, Eq a) => a -> a -> a -> Bool
isTriplet a b c = if c^2 == a^2 + b^2 then True else False
```

Q4. Function that returns all the positive integers of a list

```
posint :: [Int] -> [Int]
posint [] = []
posint (x:s)
   | x > 0 = x : (posint s)
   | otherwise = (posint s)
```

Q5. Function that counts all positive numbers in a list

```
countpos :: [Int] -> Int
countpos [] = 0
countpos (x:s)
   | x > 0 = 1 + (countpos s)
   | otherwise = (countpos s)
```

Q5. Function that returns half of each even number in a list

```
halfEvens :: [Int] -> [Int]
halfEvens x = [a `div` 2 | a <- x, a `mod` 2 == 0]
```

Q6. Function that takes a list of integers and returns a list of integers in which each odd element of the list is replaced by its square

```
squareOdds :: [Int] -> [Int]
squareOdds [] = []
squareOdds (x:s)
   | (x) `mod` 2 == 1 = x^2 : (squareOdds s)
```

```
| otherwise = x : (squareOdds s)
```