PCPF Lab
Lab Assignment number 06

Name: Aamir Ansari **Batch:** A **Roll** no. 01

Aim: To study and implement pattern Matching and Case in haskell.

Theory: Pattern matching is a mechanism for checking a value against a pattern. A successful match can also deconstruct a value into its constituent parts. It is a more powerful version of the switch statement in Java and it can likewise be used in place of a series of if/else statements.

For example, if [1, 2] is matched against [0, bot], then 1 fails to match 0, so the result is a failed match. (Recall that bot, defined earlier, is a variable bound to _|_.) But if [1, 2] is matched against [bot, 0], then matching 1 against bot causes divergence (i.e. _|_).

The other twist to this set of rules is that top-level patterns may also have a Boolean guard, as in this definition of a function that forms an abstract version of a number's sign:

```
sign x | x > 0    = 1
       | x == 0  = 0
       | x < 0  = -1
```

Pattern matching at the function head is compiled, more or less, to case expressions in the function body. Further, your use of case expressions involves pattern matching.

```
-- Pattern Matching

not  True  =  False

not False = True


-- Case Expression

not x = case x of True -> False

                  False -> True
```

PROBLEM STATEMENT 1:

Write a Haskell program to consider a function safetail that behaves in the same way as tail, except that safetail maps the empty list to the empty list, whereas tail gives an error in this case. Define safetail using:

(a) a conditional expression;

(b) guarded equations;

(c) pattern matching.

CODE:

1. Conditional expression;

```
safetail xs = if null xs
then []
else tail xs
```

2. Guarded equations;

```
safetail xs | null xs = []

| otherwise = tail xs
```

3. Pattern matching;

```
safetail [] = []
safetail xs = tail xs
```

OUTPUT:

1.

```
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> :cd C:\Users\Ashwini Parwatikar\Desktop\H34
Prelude> :load a2.hs
[1 of 1] Compiling Main             ( a2.hs, interpreted )
Ok, one module loaded.
*Main> [1,2,3,4,5]
[1,2,3,4,5]
*Main> safetail[1,2,3,4,5]
[2,3,4,5]
*Main> null[]
True
*Main>
```

2.

```
Prelude> :load a2.hs
[1 of 1] Compiling Main             ( a2.hs, interpreted )
Ok, one module loaded.
*Main> safetail[1...5]

<interactive>:8:11: error:
    Variable not in scope: (...) :: Integer -> Integer -> a
*Main> safetail[1..5]
[2,3,4,5]
*Main>
```

3.

```
*Main> :load a2.hs
[1 of 1] Compiling Main             ( a2.hs, interpreted )
Ok, one module loaded.
*Main> safetail[11,12,13,14,15]
[12,13,14,15]
*Main>
```

PROBLEM STATEMENT 2:

Write a Haskell program to implement a simple calculator. (use case statement)

CODE:

```
solveEqn = do
        putStrLn "Enter number 1 "
        input1 <- getLine
        putStrLn "Enter number 2 "
        input2 <- getLine
        let a = (read input1 :: Int)
        let b = (read input2 :: Int)
        putStrLn "Enter the operator from(+,-,/,*)"
        oper <- getChar
        return (case oper of
                '+' -> (a + b)
```

```
            '-' -> (a - b)
             '/' -> (a `div` b)
            '*' -> (a * b))

main = do
eval <- solveEqn
print ("Answer is " ++show(eval))
```

OUTPUT:

```
Prelude> :load a1.hs
[1 of 1] Compiling Main             ( a1.hs, interpreted )

a1.hs:2:1: warning: [-Wtabs]
    Tab character found here, and in 18 further locations.
    Please use spaces instead.
  |
2 |         putStrLn "Enter number 1 "
  | ^^^^^^^^
Ok, one module loaded.
*Main> :main
Enter number 1
1
Enter number 2
3
Enter the operator from(+,-,/,*)
+
"Answer is 4"
```