PPL Lab sheet-2

Name : J.Sampat Srivatsav

Roll Number : AM.EN.U4CSE19125

Batch : C.S.E - B

Q1)

```
adhinene@sampat:~$ ghci
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude> :type tail
tail :: [a] -> [a]
Prelude> :type sqrt
sqrt :: Floating a => a -> a
Prelude> :type pi
pi :: Floating a => a
Prelude> :type exp
exp :: Floating a => a -> a
Prelude> :type ^

<interactive>:1:1: error: parse error on input ‘^’
Prelude> :type (^)
(^) :: (Integral b, Num a) => a -> b -> a
Prelude> :type (/=)
(/=) :: Eq a => a -> a -> Bool
Prelude> :type (**)
(**) :: Floating a => a -> a -> a
Prelude>
```

We need not have to specify the type in the program . Haskel can understand what is the type and how to be used .

Q2)

```haskell
thrice x=[x,x,x]
sums (x : y : ys)=x : sums (x + y : ys)
sums xs = xs
```

```
Prelude> :load q2.hs
[1 of 1] Compiling Main
Ok, one module loaded.
*Main> :type thrice
thrice :: a -> [a]
*Main> :type sums
sums :: Num a => [a] -> [a]
*Main> :type map
map :: (a -> b) -> [a] -> [b]
```

```
*Main> map thrice (sums [0..4])
[[0,0,0],[1,1,1],[3,3,3],[6,6,6],[10,10,10]]
*Main>
```

Q3)

```haskell
myProduct :: Num a =>[a]->a
myProduct i=product i
```

```
Prelude> :load q3.hs
[1 of 1] Compiling Main             ( q3.hs, interpreted )
Ok, one module loaded.
*Main> myProduct[2,3,4]
24
*Main>
```

Q4)

```
*Main> :load
Ok, no modules loaded.
Prelude> :type ['a','b','c']
['a','b','c'] :: [Char]
Prelude> :type ('a','b','c')
('a','b','c') :: (Char, Char, Char)
Prelude> :type [(False,'0'),(True,'1')]
[(False,'0'),(True,'1')] :: [(Bool, Char)]
Prelude> :type [(False,True),('0','1')]

<interactive>:1:16: error:
    • Couldn't match expected type ‘Bool’ with actual type ‘Char’
    • In the expression: '0'
      In the expression: ('0', '1')
      In the expression: [(False, True), ('0', '1')]

<interactive>:1:20: error:
    • Couldn't match expected type ‘Bool’ with actual type ‘Char’
    • In the expression: '1'
      In the expression: ('0', '1')
      In the expression: [(False, True), ('0', '1')]
Prelude> :type ([False,True],['0','1'])
([False,True],['0','1']) :: ([Bool], [Char])
Prelude> :type [tail,init,reverse]
[tail,init,reverse] :: [[a] -> [a]]
Prelude>
```

Q5)

```haskell
bools :: [Bool]
bools=[False]
nums :: Int
nums=69
add :: Int -> Int -> Int -> Int
add i j k= i+j+k
copy :: a -> (a,a)
copy a=(a,a)
apply :: (a -> b) -> a -> b
apply a b=a b
```

```
Prelude> :load q5.hs
[1 of 1] Compiling Main             ( q5.hs, interpreted )
Ok, one module loaded.
*Main> type bools

<interactive>:37:11: error:
    parse error (possibly incorrect indentation or mismatched brackets)
*Main> :type bools
bools :: [Bool]
*Main> :type nums
nums :: Int
*Main> :type add
add :: Int -> Int -> Int -> Int
*Main> :type copy
copy :: a -> (a, a)
*Main> :type apply
apply :: (a -> b) -> a -> b
*Main>
```

Q6)

```
*Main> :load
Ok, no modules loaded.
Prelude> second xs=head (tail xs)
Prelude> :type second
second :: [a] -> a
Prelude> swap (x,y)=(y,x)
Prelude> :type swap
swap :: (b, a) -> (a, b)
Prelude> pair x y=(x,y)
Prelude> :type pair
pair :: a -> b -> (a, b)
Prelude> double x=x*2
Prelude> :type double
double :: Num a => a -> a
Prelude> palindrome xs=reverse xs==xs
Prelude> :type palindrome
palindrome :: Eq a => [a] -> Bool
Prelude> twice f x=f(f x)
Prelude> :type twice
twice :: (t -> t) -> t -> t
Prelude>
```

Q7)

```
always0 :: Int ->Int
always0 x=0
```

```
Prelude> :load q7.hs
[1 of 1] Compiling Main             ( q7.hs, interpreted )
Ok, one module loaded.
*Main> always0 55
0
*Main> always0 69
0
*Main>
```

Q8)

```haskell
subtracts :: Int -> Int -> Int
subtracts a b = a-b
```

```
Prelude> :load q8.hs
[1 of 1] Compiling Main             ( q8.hs, interpreted )
Ok, one module loaded.
*Main> subtracts 1 2
-1
*Main> subtracts 7 5
2
*Main> subtracts 10 0
10
*Main> subtracts 10.5 0

<interactive>:69:11: error:
    • No instance for (Fractional Int) arising from the literal ‘10.5’
    • In the first argument of ‘subtracts’, namely ‘10.5’
      In the expression: subtracts 10.5 0
      In an equation for ‘it’: it = subtracts 10.5 0
*Main>
```

It doesnot work for float .

Modified :

```haskell
subtracts :: Float -> Float -> Float
subtracts a b = a-b
```

```
*Main> subtracts 11.8 4.1
7.7000003
*Main> subtracts 11.8 4.0
7.8
*Main> subtracts 11.8 4
7.8
*Main> subtracts 110.5 4.9
105.6
*Main> subtracts 5 77
-72.0
*Main>
```

Q9)

```
addmult :: Float -> Float -> Float -> Float
addmult p q r = (p+q)*r
```

```
Prelude> :load q9.hs
[1 of 1] Compiling Main             ( q9.hs, interpreted )
Ok, one module loaded.
*Main> addmult 1 2 3
9.0
*Main> addmult 10 12 1000
22000.0
*Main> addmult 10 12.5 1000.2
22504.5
*Main>
```