Lab Exercises

1)What is the type of the following values. If the expression is invalid briefly state why.

a) [1,2,3,4]

b) [1,'2', "3"]

c) (1,2,3,4)

d) [("Yes", True), ("No", False)]

e) [[False], [True], [False,True], [True, False]]

f) [tail, reverse]

g) (head, tail, reverse)

h) ["x":[]]

i) 4==5

j) (==)

k) []

l) ()

m) show False

n) read "5" - 2

2)Give suitable polymorphic type assignments for following functions.

a) fst (x,y) = x

b) square x = x*x

c) addVectors (x1,y1) (x2,y2) = (x1 + x2, y1 + y2)

d) palindrome xs = reverse xs == xs

e) const x y = x

f) splitAt n xs = (take n xs, drop n xs)

g) apply f x = f x

h) twice f x = f (f x)

i) composition f g x = f (g x)

j) flip f x y = f y x

k) pair (f,g) x = (f x, g x)

3) Create a function **emptyList** :: [a] -> Bool that decides if a list is empty or not.

4) Define a function safetail :: [a] -> [a] that behaves in the same way as tail expect that it maps the empty list to itself rather than producing an error. (Hint: Use tail and emptyList function)
a) using conditional expression
b) using guarded equations
c) using pattern matching

5) Write a function majority which takes 3 boolean values and returns the majority among the three. For example,

>>majority True True False
True

>>majority False True False
False

6)Write a function replaceHead that takes a list and an element as argument and returns the list after replacing the head of the list with the second argument. For example,

>>replaceHead [1,2,3,4] 5

[5,2,3,4]

>>replaceHead [] 2
[]

7) Write a function curry that takes a uncurried function of 2 argument, and returns a curried version of the function. Determine appropriate type for this function.

```
plus :: Num a => (a,a) -> a
plus (x,y) = x+y

curry . . = . .
plusc = curry plus

Prelude> plusc 2 3
5
```

8) Write a function uncurry that takes a curried function of two arguments, and

returns an uncurried version of the function.Determine appropriate type for this function.

```
mult :: Num a => a -> a -> a
mult x y = x * y
```

```
uncurry .. = ..
```

```
multuc = uncurry mult
```

```
Prelude> multuc (2,3)
6
```