Exercise 1

We know that list a is an applicative:

instance Applicative [] where

pure a = [a]

fs <*> xs = [fx | f<-fs, x<-xs]

then in expression, [(+), (*), (/)] <*> [1,2,3] <*> [2,3], after the first <*> we have [(1+), (2+),(3+),(1*),(2*),(3*),(1/),(2/),(3/)] <*> [2,3,4] and the final result is [(1+2), (2+2),(3+2),(1*2),(2*2),(3*2),(1/2),(2/2),(3/2), [(1+3), (2+3),(3+),(1*3),(2*3),(3*3),(1/3),(2/3),(3/3)],where each expression like (1/3) should be evaluated. Observe that the value of (1/3) is a float 0.33333...., and hence the final list will contain all floats.

Thus, each function is applied to each element of [1,2,3] and then each function obtained this way is applied to each element of [2,3]. In other words the cartesian product of the element s of the lists is considered.

There is another interpretation of <*> that may be interesting to consider. The above expression will have the following result:

[(+), (*), (/)] <*> [1,2,3] <*> [2,3] =[1+2,2*3]

where the first function is applied to 1 and 2, the second function is applied to 2 and 3 and the third function is not applied at all since the third list is finished.

In order to define functions pure and <*> that model this behaviour of lists, we need to define a second type list that is as follows:

newtype ZipList a = Z [a] deriving Show

newtype is used for declarations with one constructor only. The definition of the fact that ZipList is Applicative is as follows:

instance Applicative ZipList **where**

pure x = ZipList (repeat x)

.... <*> ....= ----to be completed

you should define <*> in such a way that it realizes the list behaviour explained above.

Exercise 2

a) Write an Haskell program that, given three lists L1, l2 and L3 of length n >0, computes the list containing all lists of three elements where the first element comes from L1, the second element from L2 and the third from L3.

b) Write an Haskell program that, given lists L1, L2 and L3 of length n>0, computes the list containing n lists of three elements where for the i-th such list, the first element is the i-th element of L1, the second element is the i-th element of L2 and the third is the i-th element of L3.

c) Write an Haskell program that, given a list L of lists of Int, produces a new list that is obtained from L by adding 1 to the first list contained in L, then adding 2 to the second list of L ad so on.