1) Írjunk egy függvényt ami bekér két számot és visszaadja az abba az intervallumba tartozó számok összegét! (a bekért számokra nincs megkötés, hogy a kisebbet edjuk-e meg először vagy a nagyobbat)
getSum :: Int -> Int -> Int

Tesztesetek:
getSum 1 3 == 6
getSum 3 1 == 6
getSum (-3) 3 == 0
getSum 3 (-3) == 0

2) Írjunk egy függvényt amely egy listagenerátorba bekért intervallumra visszaadja egy listában, hogy mely számok prímek! (megkötjük, hogy először a kisebb számot adjuk meg)
primeList :: Int -> Int -> [Int]

Tesztesetek:
primeList 0 10 == [2,3,5,7]
primeList 10 100 == [11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]

3) Írjunk egy olyan függvényt ami bekér egy szöveget és lekódolja az alábbi módon:
Betűnként felosztja a szöveget, majd egy redezett párokat tartalmazó listába bepakolja, ahol a pár első eleme a betű a második pedig, hogy egymás után mennyi szerepel belőle!
(Ehez a feladathoz érdemes szemügyre venni a Data.Listbe található **group** függvényt és a **head** függvényt, modult importálni a file elejére beírt import kulcsszóval lehet)
encode :: String -> [(Char,Int)]

Tesztesetek:
encode “Hello” == [('H',1),('e',1),('l',2),('o',1)]
encode "AAaaBffffffCccccGGGGGGGG" == [('A',2),('a',2),('B',1),('f',5),('C',1),('c',4),('G',8)]
encode "Abbbrakkkkaaa Dabbbrraaa" ==[('A',1),('b',3),('r',1),('a',1),('k',4),('a',3),(' ',1),('D',1),('a',1),('b',3),('r',2),('a',3)]

4) Írjunk egy függvényt amia z előző kódolást megfejti nekünk! (Ehez érdemes megnézni a **replicate** valamint a **fst**, **snd** és a **concat** függvényeket)
decode :: [(Char,Int)] -> String

Tesztesetek:
decode [('H',1),('e',1),('l',2),('o',1)] == “Hello”
decode [('A',2),('a',2),('B',1),('f',5),('C',1),('c',4),('G',8)] == "AAaaBffffffCccccGGGGGGGG"
decode [('A',1),('b',3),('r',1),('a',1),('k',4),('a',3),(' ',1),('D',1),('a',1),('b',3),('r',2),('a',3)] == "Abbbrakkkkaaa Dabbbrraaa"