Functional Languages 7th practice

1. Redefine function `sum`.

```
sum' [4,5,6] == 15
sum' []      == 0
sum' [1..10] == 55
```

2. Redefine function `last`, which is undefined for empty list.

```
last' [4,5,9]  == 9
last' [5]      == 5
last' [1..100] == 100
```

3. Redefine function `and`.

```
and' [True, True, True, True]
and' []
not (and' [True, False, True, True])
not (and' [False, False, False])
```

4. Redefine function `or`.

```
or' [True, True, True, True]
or' [True, True, False, True]
not (or' [])
not (or' [False, False, False, False])
```

5. Redefine function `replicate`.

```
replicate' 5 'a'  == "aaaaa"
replicate' 10 'p' == "pppppppppp"
```

6. Extend a string to a specific width. If the string has more characters than the specified width then leave the string unchanged.

```
format 10 "haskell" == "  haskell"
format 3 "haskell"  == "haskell"
```

7. Insert an integer into a list in order. It should follow smaller integers and precede greater ones.

```
insert 2 [4, 5, 9]  == [2, 4, 5, 9]
insert 6 [4, 5, 9]  == [4, 5, 6, 9]
insert 11 [4, 5, 9] == [4, 5, 9, 11]
insert 5 [4, 5, 9]  == [4, 5, 5, 9]
insert 2 []         == [2]
```

8. Implement insertion sort, which sorts by inserting integers one after the other in a list in order.

```
sort [5, -2, 3] == [-2, 3, 5]
sort []         == []
```

9. *Recursively merge two sorted list of integers so the result list is also sorted.

```
merge [4, 5, 7, 10] [2, 3, 6] == [2, 3, 4, 5, 6, 7, 10]
merge [5] [1, 2, 8]           == [1, 2, 5, 8]
merge [2, 3] []               == [2, 3]
```

10. *Implement merge sort, which splits a list in half then sorts the halves then merges them.

```
mergeSort [5, 2, -4, 9, 3] == [-4, 2, 3, 5, 9]
mergeSort []               == []
```

11. Define a function `breakOn` which returns parts of a string before and after a character.

```
breakOn ' ' "haskell is cool"      == ("haskell", " is cool")
breakOn ' ' "is cool"              == ("is", " cool")
breakOn '/' "haskell/gyak/gyak.hs" == ("haskell", " gyak/gyak.hs")
```

12. Define a function `splitOn` which splits a list at each occurence of a character.

```
splitOn ' ' "haskell is cool"      == ["haskell", "is", "cool"]
splitOn '/' "haskell/gyak/gyak.hs" == ["haskell", "gyak", "gyak.hs"]
splitOn '/' "haskell"              == ["haskell"]
splitOn '/' ""                     == []
```

13. Split a csv file into list of cells. Function `lines` is helpful in taking lines of a string.

```
csv "nev,neptun,jegy\nEndre,ADG1K5,5\nAnna,KRJ25L,5"
  == [["nev","neptun","jegy"],["Endre","ADG1K5","5"],["Anna","KRJ25L","5"]]
```