Exercises

1) Create a function which constructs a new list by adding two lists by adding their corresponding elements. For example, List(3,0,1) and List(4,3,2) become List(7,3,3). Generalize the function so that it is not specific to integer or addition

```
def combinelist[A](f:(A,A)=>A ,list1 : List[A], list2 : List[A]):List[A]= {
 if (list1.nonEmpty) {
   List(f(list1.head, list2.head)) ::: combinelist(f, list1.tail, list2.tail)
 }else Nil
}
```

2) Define a Scala procedure “filter-list”, which takes a predicate and a list as arguments, and returns a list that contains the elements of the given list that satisfy the given predicate (or condition). Use this to remove all even numbers from a list

```
def filter[A](f:(A)=>Boolean,list : List[A]):List[A]={
 if(list.nonEmpty){
   if(f(list.head)){
     list.head::filter(f,list.tail)
   }else {
     filter(f,list.tail)
   }
 }else {
   Nil
 }
}
```

3) Concatenating a list of lists into single list

```
def filter[A](f:(A)=>Boolean,list : List[A]):List[A]={
 if(list.nonEmpty){
   if(f(list.head)){
     list.head::filter(f,list.tail)
   }else {
     filter(f,list.tail)
   }
 }else {
   Nil
 }
}
```

4) Check if a given list is sorted

def isSorted[A](as: List[A], ordering: (A,A) => Boolean,int index): Boolean = as match {

```
            Case x::Nil => True
            Case x::middle::xs => ordering(x,middle) match {
                        Case True => isSorted(as,ordering,middle::xs)
                        Case _ => False
}

}
```

```
def isSorted[A](as:List[A], ordering: (A, A) => Boolean,int index): Boolean =  {

@annotation.tailrec
  def go(n: Int): Boolean =
            if (n >= as.length - 1) true
            else if (ordering(as(n), as(n + 1))) false
            else go(n + 1)

  go(0)
}
```

5) Recursive quicksort

```
def quickSort[T](xs: List[T])(p: (T, T) => Boolean): List[T] = xs match{
    case Nil => Nil
    case _ =>
         val x = xs.head
         val (left, right) = xs.tail.partition(p(_, x))
         val left_sorted = quickSort(left)(p)
         val right_sorted = quickSort(right)(p)
         left_sorted ::: (x :: right_sorted)
}
```