Exercise 8.1

For the character list type \textit{CharList} from the lecture, write both recursive and iterative functions that perform the following tasks:

(a) Calculate the length of a list.
(b) Duplicate each list element, thus turning for example “abcd” into “aabbccccddd”.
(c) Given two characters $x \leq y$, produce a list containing in sequence all characters from $x$ to $y$ inclusively.
(d) Produce a copy of a list.
(e) Reverse a list.

Exercise 8.2 — Textbook Insertion

Read and understand the textbook version of insertion into lists (\textit{fig12_03.c}).
Manually simulate appropriate test cases.

Exercise 8.3 — Calendar (ctd.)

For the calendar application of Exercise 6.2, adapt the \textit{Day} data type to allow an arbitrary number of appointments, and adapt your \textit{find} function accordingly.
One aspect to keep in mind is that it should be reasonably easy to add and delete single appointments.

Exercise 8.4 — Number Lists (51\% of Midterm 3, 2005)

The following C type definitions will be used to define “number lists” as singly-linked lists of \texttt{int} elements:

```
typedef struct NumListNodeStruct {
    int elem;
    struct NumListNodeStruct * next;
} NumListNode;
```

typedef NumListNode * NumList;

The considered number lists will always have their elements in \textbf{ascending order}.

(The items are \textit{independent of each other}!)

(a) $\approx 12\%$ \textbf{Implement} the summing up of all the \texttt{elements} in a list.
Define \textbf{two versions}: one \texttt{recursive} and one \texttt{iterative} function.
\textbf{Document} the function interface!

(b) $\approx 39\%$ \textbf{Design} and \textbf{implement} a function that splits a list into two sub-lists, one containing all the even numbers from the original list, and the other all the odd numbers from the original list (both in ascending order). \textbf{Carefully document the function interface}. 