A note on storage management in Lab assignment 1

Dynamic allocation/de-allocation of storage, in general, is a subject that is beyond the scope of this course. In this note I will confine to a possible solution that is specific to the current lab assignment.

When two lists are merged, a new list gets created that needs storage allocation. Can we reclaim that storage? The answer is yes. Since the operand lists are no longer required after forming the merged list, we can overwrite the operand lists by the merged list. This is doable because -

a) size of merged list ≤ size of operand list 1 + size of operand list 2,
AND
b) operand lists are always in contiguous space, since these are formed out of the same list.

Note that we are talking of lists of pointers to the strings.

Here are two implementations. In the first one, space allocation/de-allocation is done by the caller. In the second one, space allocation/de-allocation is done by the callee. In both cases, the function sort can be seen as performing sorting operation 'in-place', that is, sorted list replaces the given list.

Implementation 1

```
function sort (list)
   let list1 and list2 be two parts of list
   sort (list1)
   sort (list2)
   allocate space for newlist on stack
   merge (list1, list2, newlist)
   copy newlist onto list
   dispose off space allocated to newlist
end
```

```
function merge (list1, list2, newlist)
   merge list1 and list2 forming newlist
end
```

Implementation 2

```
function sort (list)
   let list1 and list2 be two parts of list
   sort (list1)
   sort (list2)
   merge (list1, list2)
end
```

```
function merge (list1, list2)
   allocate space for newlist on stack
   merge list1 and list2 forming newlist
   copy newlist onto list1, list2
   dispose off space allocated to newlist
end
```