Algorithms II
Autumn Semester 1998
Coursework Exercise 1

This work must be handed in by 4:00pm on **13th. November 1998**. This work will contribute 10% towards your final mark for this module.

---

1. Write an algorithm, *length(list)*, in SPARKS pseudocode to count the number of nodes in a singly-linked list *list*, where *list* points to the head node of the list. The head node itself should *not* be counted. Assume the last node has link field set to *end_of_list*.

2. (a) Describe fully an example of a recursive algorithm **not** discussed in the lecture notes.

   (b) Consider a coin set containing coins with the following values: 1, 5, 10, 20, 25. Does the greedy algorithm described in the lecture notes always work optimally for this coin set? Justify your answer.

   (c) Draw a tree to represent a binary search of the following sorted list of integers;

   1, 3, 5, 8, 11, 14, 15, 19, 21, 22, 24, 28, 30, 32, 33

3. This question is concerned with using a doubly-linked, circular list for keeping track of free memory on multiprocessing workstations, as discussed in the lectures and in section 4.8 of Horowitz and Sahni.

   (a) What are the four possible cases that can arise when an allocated block of memory is returned to the free list if a coalescing strategy is used?

   (b) The following segment of SPARKS pseudocode describes how coalescing is performed when the memory block to the right of the freed block is also free, but the block to the left is not. p points to the memory being freed and n is its size.

1. *rlink(llink(p+n)) ← p*
2. *llink(rlink(p+n)) ← p*
3. *llink(p) ← llink(p+n)*
4. *rlink(p) ← rlink(p+n)*
5. *size(p) ← n + size(p+n)*
6. *uplink(p+size(p)-1) ← p*
7. *tag(p) ← 0*
8. **if** *av* = *p* + *n* **then** *av ← p*

i. Describe clearly in words what lines 1 and 2 of the above pseudocode do.

ii. Describe clearly in words what lines 3 and 4 of the above pseudocode do.

iii. What is *uplink*?

iv. Why is *uplink* needed in the node structure of a free node?

v. In line 7 of the above pseudocode, the tag value at the start of the freed block is set to zero. Why does the pseudocode not also set the tag value at the end of the free block to zero?

vi. Why is line 8 of the above pseudocode necessary?

(c) The following SPARKS pseudocode describes the case when the blocks to the left and the right of the freed block are free, and must be coalesced with the freed block.

rlink(llink(p+n)) ← rlink(p+n)
llink(rlink(p+n)) ← llink(p+n)
q ← uplink(p-1)
size(q) ← size(q) + n + size(p+n)
uplink(q+size(q)-1) ← q
if *av* = *p* + *n* **then** *av ← llink(p+n)*

Describe clearly in words what this fragment of pseudocode does.