Exam Algorithms and Data Structures in C

11 April 2014, 9 - 12 h.

This exam contains 4 problems, yielding in total 90 points.
The exam grade is (# points)/10 + 1.

1. (25 point)
   This problem is about binary trees defined by the following type definition:

```
typedef struct TreeNode *Tree;

struct TreeNode {
  int item;
  Tree leftChild, rightChild;
};
```

   (a) When is a binary tree a search tree?

   (b) Define the C function with prototype

```
Tree addInSearchTree(Tree t, int n);
```

   that adds `n` to search tree `t` (provided `n` does not occur in `t`) while preserving the search tree property. When `n` occurs in `t`, the returned tree is equal to the input tree.

   (c) Define the C function with prototype

```
Tree removeFromSearchTree(Tree t, int n);
```

   that removes `n` from `t` (provided `n` occurs in `t`) while preserving the search tree property. When `n` does not occur in `t`, the returned tree is equal to the input tree. You may *use* the function with prototype

```
int successor(Tree t);
```

   (you do not have to *define* this function). Precondition for the function `successor` is that `t` has a right child. The function `successor` returns the smallest integer `m` in the subtree that has the right child of `t` as root, and it removes the node containing `m`.

2. (20 point)
The C code below defines types and functions for the implementation of lists of integers. However, there are 4 errors in the code so that functions do not work properly and/or memory leaks may occur. Find these errors, indicate what is wrong and repair them.

```
1  typedef struct ListNode *List;
2
3  struct ListNode {
4    int item;
5    List next;
6  };
7
8  List addItem(int n, List li) {
9    List newList = malloc(sizeof(struct ListNode));
10   assert(newList!=NULL);
11   newList->item = n;
12   newList->next = li;
13   return newList;
14 }
15
16 List removeFirstNode(List li) {
17   List returnList;
18   if ( li == NULL ) {
19     printf("list␣empty\n");
20     abort();
21   }
22   returnList = li->next;
23   free(li);
24   return returnList;
25 }
26
27 List insertInOrder(List li, int n) {
28 /* li is sorted in ascending order */
29   List li1;
30   if ( li->item > n || li == NULL  ) {
31     return addItem(n,li);
32   }
33   li1 = li;
34   while ( li1->next != NULL && (li1->next)->item < n ) {
35     li1 = li1->next;
36   }
37   return li;
38 }
```

```
39 int removeLastOcc(List *lp, int n) {
40 /* NB: lp is a reference pointer!
41  * the function removes the last occurrence of n from *lp
42  * it returns 1 when an occurrence of n has been removed,
43  * otherwise 0
44  */
45   if ( *lp == NULL ) {
46     return 0;
47   }
48   if ( (*lp)->item == n ) {
49     *lp = removeFirstNode(*lp);
50     return 1;
51   }
52   if ( removeLastOcc(&((*lp)->next),n) ) {
53     return 1;
54   }
55   return 0;
56 }
57
58 List removeAllOcc(List li, int n) {
59 /* remove all occurrences of n and return the resulting list */
60   if ( li == NULL ) {
61     return NULL;
62   }
63   if ( li->item == n ) {
64     return removeAllOcc(li->next,n);
65   } else {
66     li->next = removeAllOcc(li->next,n);
67     return li;
68   }
69 }
```

3. (25 points)
   This problem is about tries.

   (a) Let W be a collection of words. Define: T is a standard trie for W.

   (b) Describe in pseudocode an algorithm to search for a word in a trie.

   (c) Explain what a suffix trie is, and how it can be used to search for a pattern in a text.

4. (20 points)
Consider the following algorithm:

```
algorithm BreadthFirstSearch(G,v)
    input connected graph G with node v;
        all nodes and edges ane unlabeled
    result labeling of the edges of G with NEW and OLD;
        the edges with label NEW form a spanning tree of G,
        and all nodes are visited (and labeled VISITED)
    give v the label VISITED
    create an empty queue Q
    enqueue(v)
    while Q not empty do
        u ← dequeue()
        forall e incident with u do
            if e has no label then
                w ← the other node incident with e
                if w has no label then
                    give e the label NEW
                    give w the label VISITED
                else
                    give e the label OLD
```

(a) What is a *spanning tree* of a connected graph?

(b) The algorithm contains one error. Indicate what the error is and repair it.

(c) Modify the corrected algorithm into an algorithm FindPath(G,v,w) that finds a path from v to w in graph G.

(d) Argue that the path found by FindPath has minimal length. Here the length of a path is the number of edges in it.