Exam Algorithms and Data Structures in C

Thursday 8 May 2014, 18:30 - 21:30 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 where each node contains an integer.

   (a) Give a type definition for the type `Tree` of binary trees.

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

   (c) Define the C function with prototype

   ```
   Tree makeSearchTree(int ar[], int n);
   ```

   that constructs a balanced search tree from the integers in the array `ar` of length `n`. You may assume that `ar` is sorted in ascending order, and that all integers in `ar` are different.

   (d) Define the C function with prototype

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

   that counts the number of nodes in `t` that contain an integer $< n$.

2. (20 point)
The C code below defines types and functions for the implementation of queues by lists. 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  typedef struct Queue {
9    List list;
10   List lastNode;
11 } Queue;
12
13 void listEmptyError() {
14   printf("list empty\n");
15   abort();
16 }
17
18 List addItem(int n, List li) {
19   List newList = malloc(sizeof(struct ListNode));
20   assert(newList!=NULL);
21   newList->item = n;
22   newList->next = li;
23   return newList;
24 }
25
26 int firstItem(List li) {
27   if ( li == NULL ) {
28   listEmptyError();
29   }
30   return li->item;
31 }
32
33 List removeFirstNode(List li) {
34   if ( li == NULL ) {
35     listEmptyError();
36   }
37   free(li);
38   return li->next;
39 }
40
41 void freeList(List li) {
42   if ( li == NULL ) {
43     return;
44   }
45   freeList(li->next);
46   return;
47 }
```

```
48 Queue newEmptyQueue () {
49   Queue q;
50   q.list = NULL;
51   q.lastNode = NULL;
52   return q;
53 }
54
55 int isEmptyQueue (Queue q) {
56   return (q.list == NULL);
57 }
58
59 Queue enqueue (int n, Queue q) {
60   if ( q.list == NULL ) {
61     q.list = addItem(n,NULL);
62     q.lastNode = q.list;
63   } else {
64     (q.lastNode)->next = addItem(n,NULL);
65   }
66   return q;
67 }
68
69 int dequeue(Queue *qp) {  /* precondition: qp != NULL */
70   int n = firstItem(qp->list);
71   qp->list = removeFirstNode(qp->list);
72   return n;
73 }
74
75 void freeQueue (Queue q) {
76   freeList(q.list);
77 }
```

3. (25 points)
This problem is about heaps containing integers.

   (a) Give a definition of a heap.

   (b) Explain how a heap can be represented by an array of integers.

   (c) Describe in pseudocode the algorithm Enqueue to add an integer to a heap, and also the auxiliary algorithm Upheap to restore heap order. You may use either the array representation or the pointer representation of heaps.

4. (20 points)
Consider the following algorithm:

```
algorithm Dijkstra (G, v)
    input connected weighted graph G with node v;
        the weights are nonnegative
    output function d yielding for every node the length of a shortest path to v
    S ← nodes(G)
    forall u ∈ nodes(G) do
        d[u] ← ∞
    while S is not empty do
        u ← node in S with minimal value of d
        forall z ∈ S with (u,z) ∈ edges(G) do
            d[z] ← min(d[z], d[u])
    return d
```

(a) The algorithm contains three errors. Indicate what the errors are and repair them.

(b) Modify the corrected algorithm into an algorithm FindShortestPath(G,v,w) that finds and returns a shortest path from v to w in graph G.