Exam 2

April 25th, 2018

Work alone. Do not use any notes or books. You have approximately 75 minutes to complete this exam.

Please write your answers on the exam. More paper is available if you need it. Please put your name at the top of the first page.

There are four questions on this exam, for a total of 80 points.

1 Restructuring a binary search tree (20 points)

Below is a picture of a perfectly balanced binary search tree.

Show that it is possible to move `3` to the root using at most two tree rotations. Draw the tree after each rotation.

```
      4
     / \
    2   6
   /\   /\
  1  3 5  7
```

Solution

```
      4
     / \
    3   6
   /    /\
  2    5  7
 /
1
```

```
      3
     / \
    2   4
   /     \
  1       6
         /\
        5  7
```

2 Sorting (20 points)

Write a function that takes an argument n and an array a of n `int`s, as declared below, and returns 1 if the array is sorted in strictly increasing order, and 0 otherwise. The array is strictly increasing if $a[i] < a[j]$ whenever $i < j$. Your function should run in $O(n)$ time.

```
#include <stdlib.h> // for size_t
int isSorted(size_t n, const int a[]);
```

Solution

```
int isSorted(size_t n, const int a[])
{
    for(size_t i = 0; i+1 < n; i++) {
        if(!(a[i] < a[i+1])) {
            return 0;  // not sorted
        }
    }
    return 1;
}
```

3 Short paths (20 points)

Write a function that takes a graph represented in adjacency-list form as declared below, and returns 1 if the graph contains a path of length exactly 2 from vertex `u` to vertex `v`, and 0 otherwise. (Such a path consists of two edges, one from `u` to some intermediate vertex `w`, and one from `w` to `v`.)

```
struct edge { struct edge *next; int destination; };
struct graph { int numVertices; struct edge **adjacencyLists; };
int hasPathOfLength2(const struct graph *g, int u, int v);
```

Solution

```
int hasPathOfLength2(const struct graph *g, int u, int v)
{
    for(struct edge *e1 = g->adjacencyLists[u]; e1 != 0; e1 = e1->next) {
        for(struct edge *e2 = g->adjacencyLists[e1->destination];
                e2 != 0; e2 = e2->next) {
            if(e2->destination == v) {
                return 1;
            }
        }
    }
    return 0;
}
```

4 A woolly tree (20 points)

What output does the following program produce? Please draw a rectangle around your answer to distinguish it from any notes you might make.

```
#include <stdio.h>
struct t { int k; struct t *c[2]; };
void show(const struct t *x) {
    if(x) {
        putchar(x->k);
        for(int i = 0; i < 2; i++) { show(x->c[i]); }
    }
}
#define N (5)
int main(int argc, char **argv) {
    struct t a[N];
    for(int i = 0; i < N; i++) {
        a[i].k = 'a' + i;
        a[i].c[0] = a + i/2;
        a[i].c[1] = a + i/3;
    }
    a[0].c[0] = a[0].c[1] = 0;
    show(a+N-1);
    return 0;
}
```

Solution

`ecbaaabaa`