Exam 1

March 1st, 2012

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 Not always a heap (20 points)

Suppose we have a binary tree built from nodes declared as follows:

```
struct node {
    struct node *child[2];
    int key;
};
```

Write a function `checkHeap` that returns `1` if its argument satisfies the max-heap property (each node's key is greater than or equal to the keys of its children) and `0` otherwise. This function is declared as follows:

```
int checkHeap(struct node *h);
```

Clarifications added during exam: The argument `h` is the root of a heap (or `0` if the heap is empty); your function should check every node in the heap. You may assume the heap is not too deep. The empty heap satisfies the heap property.

Solution

```
int checkHeap(struct node *h)
{
    if(h == 0) {
        /* empty heap has the heap property */
        return 1;
    } else if(h->child[0] != 0 && h->child[0]->key > h->key) {
        /* child[0] is too big */
        return 0;
    } else if(h->child[1] != 0 && h->child[1]->key > h->key) {
        /* child[1] is too big */
        return 0;
    } else {
        /* we are OK, check both subtrees */
        return checkHeap(h->child[0]) && checkHeap(h->child[1]);
    }
}
```

2 Record-setting (20 points)

Write a function that computes the number of *left-to-right maxima* in an array of `int`s, where a left-to-right maximum is a value that is greater than any value at an earlier position in the array. For example, the array `0 1 2` has three left-to-right maxima, while the array `4 1 4 5 3 2` has only two (the first `4` and the `5`).

Your function should be as efficient as you can make it. The declaration of the function is:

```
/* return the number of left-to-right maxima */
/* in array a of length n */
int leftToRightMaxima(int n, const int a[]);
```

Solution

```
int leftToRightMaxima(int n, const int a[])
{
    int i;
    int max;      /* max seen so far */
    int count;

    if(n == 0) { return 0; }

    max = a[0];
    count = 1;

    for(i = 1; i < n; i++) {
        if(a[i] > max) {
            max = a[i];
            count++;
        }
    }

    return count;
}
```

3 A buggy destructor (20 points)

Here is a buggy function that attempts to free every element of a singly-linked list. Describe as many bugs as you can find in this function, and provide a new function that frees all the elements correctly.

Clarification added during exam: The uselessness of the linked list is not a bug.

```
struct elt {
    struct elt *next;
};

void listDestroy(struct elt *list)
{
    struct elt e;

    for(e = list; e != 0; e = e->next) {
        free(e);
    }

    return 0;
}
```

Solution

1. The variable `e` should be declared as `struct elt *`, not `struct elt`.
2. The `for` loop uses `e->next` after `e` has been freed.
3. The `return` statement is an error in a function with `void` return value.

Here is an improved version:

```
void listDestroy(struct elt *list)
{
    struct elt *next;

    while(list) {
        next = list->next;
        free(list);
        list = next;
    }
}
```

4 A self-referencing array (20 points)

What output is produced by the following program? Draw a rectangle around your solution, so we can distinguish it from any notes you might make.

```
#include <stdio.h>

#define N (6)

int
main(int argc, char **argv)
{
    int i;
    int a[N];
    int *p;

    for(i = 0; i < N; i++) { a[i] = i; }
    for(i = 0, p = a; i < N/2; i++, p++) { *p++ = a[i]; }
    for(i = 0; i < N; i++) { printf("%d ", a[i]); }
    putchar('\n');

    return 0;
}
```

Solution

```
0 1 1 3 1 5
```