992 Data Structures and Algorithms

Midterm

April 29, 2011

1. (15%) **Write a recursive function** that returns the largest item in a given linked list (in which each node contains a positive integer), and has all the nodes with that value removed from the list.
2. (15%) Please show the idea about how the calculation of the n-th Fibonacci number can be achieved faster than $O(n^2)$.
3. (15%) **Write a function** that frees the nodes in even positions in a given linked list (the second, fourth, sixth, and so forth)
4. (10%) Show that the "weighted quick-union" algorithm for the *connectivity* problem is O(M lg$N$), where M denotes the number of pairs on N objects.
5. (10%) **Write a program** that reads in a sequence of characters, and determines whether its parentheses, braces, and curly braces are "balanced."
6. (15%)

Give an $O(nt)$ algorithm for the following task.

> *Input:* A list of n positive integers $a_1, a_2, \ldots, a_n$; a positive integer t.
> *Question:* Does some subset of the a_i's add up to t? (You can use each a_i at most once.)

7. (20%)

A *contiguous subsequence* of a list S is a subsequence made up of consecutive elements of S. For instance, if S is

$$5, 15, -30, 10, -5, 40, 10,$$

then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not. Give a linear-time algorithm for the following task:

> *Input:* A list of numbers, $a_1, a_2, \ldots, a_n$.
> *Output:* The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero).

For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55.

(a) Solve this problem by using **bottom-up dynamic programming**.

(b) Solve this problem by using **top-down dynamic programming**.

Note: if you need to use stack ADT, use the following codes as interface:

```
template <class Item>
class STACK
  {
    private:
      // Implementation-dependent code
    public:
      STACK(int);
      int empty() const;
      void push(Item item);
      Item pop();
  };
```