CSE 326 – Data Structures and Algorithms
Autumn 2003
Dry assignment #2
Due date: 1/23/03

1. A store owner wants to keep some data about his customers. There are at most N customers, each having a unique id in {1, 2,… , N}. Suggest a data structure that supports the following operations:

`init(N)` – initializes the number of potential customers to N; no purchases are done yet by any customer.

`purchase(i,x)` – the customer having id i purchases goods of total value x. If this is the first time that i purchases anything, then i joins the store's club.

`sum(i)` - returns the total value of goods purchased by customer i so far;

`client(k)` – returns the id of the k-th customer to join the stores's club (or 0, if no customers have joined yet or if k is larger than the number of customers in the club).

For example, after performing `init(20)`, `purchase(1,1200)`, `purchase(7,100)`, `purchase(7,300)`, and `purchase(5,300)`, `client(2)` should return 7; `client(3)` should return 5; `sum(7)` should return 400; `sum(2)` should return 0; and `client(6)` should return 0.

The operations `purchase()`, `sum()`, and `client()` should have time complexity O(1). The `init()` operation is allowed to have time complexity O(N). The space complexity of your data structure should be O(N).

No need to write pseudo-code, only explain clearly in writing your data structure and the way the operations are performed.

2. An **adaptive linked list** is a regular linked list, with a header, in which whenever we insert a new element or access (via find) an existing element, it is moved to the first position in the list (after the header). The order of the other elements remains unchanged.

a. For the linked list below, draw the resulting list after the following operations are performed: `find(3)`, `insert(6)`, `delete(2)`, `find(4)`.

header - > 1 - > 2 - > 3 - > 4 - > 5

b. Implement (write pseudo-code) the operations `insert(x)`, `find(x)`, and `delete(x)` in an adaptive linked list.

c. Assume that an adaptive linked list includes a popular element, z, such that on average ¼ of the find operations are `find(z)`. In other words, out of any k find operations ¼ k are `find(z)` and ¾ k are `find()` of other elements.

Given that the list has N elements and that k=Ω(N) find operations were performed, prove that the average time of `find(z)` is O(1).

3. Consider a string that includes the characters ( ) [ ] { }. The string is called 'legal' if the characters appear in a way that obeys the following rules:

(i) every starting symbol (, [or { must have its matching end symbol),] or };
(ii) every end symbol must come after its matching start symbol (e.g.] [is illegal);
(iii) pairs must be closed off in the reverse order in which they were started (e.g. ([{] }) is illegal);
(iv) pairs may be nested inside each other (e.g. [[( )]] is legal);
(v) a brace pair { } may only be nested within another brace pair, not within a parentheses or bracket pair.

Your goal is to design a program that gets as input a string and determine if it is legal. You can assume (without checking) that no other characters appear in the string.

a) Which data structure would you use in this program? Motivate your choice in terms of the problem definition above. Why is your choice appropriate?

b) What sequence of operations would you need to perform for each of the six symbols above when they appear during your scan of the input file? Express the operations in terms of ADT calls on the data structure you chose above, plus any necessary code (or variables) to go with it. Give pseudocode; this shouldn't require more than a few lines each. If the input is illegal, you must report which rule is violated, and the specific characters in the input that were illegal.

c) For each of the six symbols that you handled in part b, explain how your pseudocode fragment upholds all 5 rules above. You don't have to give a formal proof, but you need to provide a convincing explanation of why each step you take is correct and why every possible illegality is caught and noted by your algorithms.