CSE 373 – Data Structures and Algorithms
Autumn 2003
Dry assignment #2
Due date: 10/17/03 (see submission instructions on the course web page).

1. A 'frame matrix' is an N*N matrix in which all the values along the same frame are identical. For example, the matrix below is a 5*5 frame matrix.

7	7	7	7	7
7	14	14	14	7
7	14	-9	14	7
7	14	14	14	7
7	7	7	7	7

Suggest a data structure for storing a frame matrix, whose space complexity is O(N) (for an N*N frame matrix with N^2 elements). Using your suggested data structure, implement (write in pseudo-code) the following operations; each should have time complexity O(1):

`get(i,j)` returns the value of the element whose location is (i,j).

`put(i,j,x)` – set the value x at location (i,j) AND in all the locations in the frame to which (i,j) belongs, in a way that the resulting matrix is still a frame matrix.

2. 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 customer'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 customer'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.

3. t is a linked list. What is the result of executing `rec_func(t)` (defined below)? What is its time and space complexity?

```
rec_func1(t node_pointer, r node_pointer): node_pointer
{
    tail node_pointer;
    if (t = NULL)  return r;
    tail := t.next;
    t.next := r;
    return rec_func1(tail, t);
}

rec_func(t node_pointer): node_pointer
{
      return rec_func1(t, NULL);
}
```

4. 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).