CS106X Handout #21
Spring 99-00 April 24, 2000

Section Exercises 4: 106A roundup and Linked Lists

Problem 1 : Linked list concepts
a) What is wrong with this recursive declaration attempt? How would you fix it?

```
typedef struct {
   char data;
   Cell *next;
} Cell;
```

b) Trace through the following code and draw the state of memory at the end. (This is using the typedef from below)

```
Node *p;

p = New(Node *);
p->value = 1;
p->next = New(Node *);
p = p->next;
p->value = 2;
p->next = New(Node *);
p->next->next = p;
p->next->next->next->value = 3;
```

Problem 2 : Simple linked list functions
For the linked list questions, use this type declaration for each node in the list:

```
typedef struct _Node{
   int value;
   struct _Node *next;
} Node;
```

a) Given a linked list and a value, write a function **CountOccurrences** that returns the number of times that value appears in the list.

b) Write a function **Stutter** that given a linked list will duplicate every cell in the list. If the incoming list is (11 5 21 7 7), the function will destructively modify the list to contain (11 5 5 21 21 7 7 7 7).

c) Write a function **RemoveDuplicates** that given a linked list will remove and free the second of all neighboring duplicates found in the list. If the incoming list is (5 5 22 37 89 89 15 15 22) the function will destructively modify the list to contain (5 22 37 89 15 22). Don't worry about handling duplicate sequences longer than 2 or duplicates that aren't right next to each other in the list.

Problem 3: Life File Reading

Sometimes, random seeding in Life creates interesting repeating patterns. These patterns are difficult to capture, repeat, or study later, because they are randomly generated. Write a function called `SaveSimulationAs` which captures a configuration of a Life grid and writes it to a file of the given name, so the simulation can be opened later and started at the current point. Also, write the companion function `OpenSimulationSavedAs`, which reads in a saved simulation file and configures the grid appropriately.

Problem 4: Using structs and file I/O

You have just been hired by Interpol's International Terrorism branch. As part of your first project, you have been asked to read in the latest available information on the most wanted terrorists. The information has been given to you in a text file, formatted as follows: the first line contains the number of terrorists described in the file, followed by a blank line, followed by the information on each terrorist. The information for each terrorist includes their name, their alias, their height and weight, the number of crimes they are wanted for and a list of those crimes, each on a separate line. For each crime, there is a description, the location where the crime took place and the year it occurred. The information for each terrorist ends with a blank line. You can assume that you will be reading in at most 10 terrorists from the file. There is no limit to how many crimes each terrorist can be wanted for. A sample data file looks like this:

```
2                 // Number of terrorists

Joe Schmoe        // Name of terrorist 1
Sly               // Alias
1.80              // Height (m)
76                // Weight (kg)
1                 // Number of crimes wanted for
Hacking           // Crime 1 description
Palo Alto         // Crime 1 location
1987              // Crime 1 year

Jenny Smith       // Name of terrorist 2
Ace               // and so on
1.67
52
2
Gambling
Nevada
1991
Hijacking
Chicago
1984
```

Design a data structure to store the information in the file. Your data structure should be as simple and memory-efficient as possible. *Hint: you will probably want to use dynamic allocation. Also write a function that reads in the information from the file into your data structure.*