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

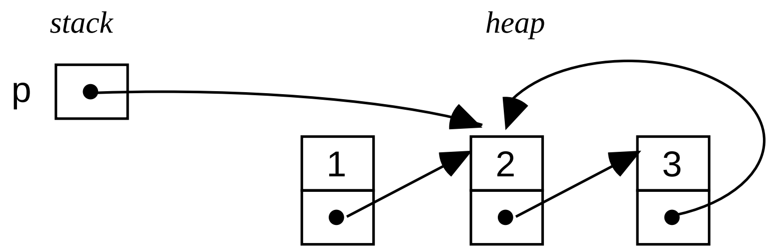
Section Solutions 4: Linked Lists

Problem 1 : Linked list concepts

a) A C compiler reads a program from top to bottom so when it encounters the field `next` declared as a `Cell *`, it doesn't yet know what a `Cell` is. By adding a tag name after the `struct` keyword, you can refer to the structure in progress as type `struct <tagname>`. By convention, tag names are usually an underscore followed by the name of the structure. Here is a working declaration:

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

b)



Problem 2 : Simple linked list functions

a)
```
static int CountOccurrences(Node *list, int value)
{
   int count;
   Node *cur;

   count = 0;
   for (cur = list; cur != NULL; cur = cur->next)
          if (cur->value == value) count++;
   return count;
}
```

b)
```
static void Stutter(Node *list)
{
   Node *cur, *newOne;

   for (cur = list; cur != NULL; cur = cur->next) {
          newOne = New(Node *);       // make new copy of current cell
          newOne->value = cur->value;
          newOne->next = cur->next;  // splice new cell after cur
          cur->next = newOne;
          cur = newOne; // move past new one so don't duplicate again
   }
}
```

c)
```
static void RemoveDuplicates(Node *list)
```

```
{
   Node *cur, *duplicate;

   for (cur = list; cur != NULL; cur = cur->next) {
         if (cur->next != NULL && cur->value == cur->next->value) { //
match!
               duplicate = cur->next;           // record ptr to duplicate
one
               cur->next = cur->next->next;     // splice it out
               FreeBlock(duplicate);            // free storage
         }
   }
}
```

Problem 3: Life File Reading

`SaveSimulationAs` and `OpenSimulationSavedAs` can be written as follows:

```
void SaveSimulationAs (gridADT grid, string filename)
{
   FILE *lifeFile;
   int i, j;

   lifeFile = fopen(filename, “w”);
   if (lifeFile == NULL) Error (“Could not open file named %s”, filename);

   fprintf(lifeFile, “%d ”, NumRows(grid));
   fprintf(lifeFile, “%d \n”, NumCols(grid));

   for (i=0; i < NumRows(grid); i++) {
         for (j=0; j < NumCols(grid); j++) {
               fprintf(lifeFile, “%d ”, GetValueAt(grid, i, j));
         }
         fprintf(lifeFile, “\n”);
   }
   fclose (lifeFile);
}
void OpenSimulationSavedAs (string filename)
{
   FILE *lifeFile;
   gridADT grid;
   int i,j,tmp,rows,cols;

   lifeFile = fopen(filename, “r”);
   if (lifeFile == NULL) Error (“Could not open file named %s”, filename);

   fscanf(lifeFile, “%d ”, &rows);
   fscanf(lifeFile, “%d \n”, &cols);

   grid = CreateGrid(rows, cols);

   for (i=0; i < NumRows(grid); i++) {
         for (j=0; j < NumCols(grid); j++) {
               fscanf(lifeFile, “%d ”, &tmp);
               SetValueAt(grid,tmp,i,j);
         }
         fscanf(lifeFile, “\n”);
   }
   fclose (lifeFile);
}
```

Problem 4: Using structs and file I/O

Because there will be at most 10 terrorists to read in, the most efficient data structure is just an array of `MAX_TERRORISTS` pointers to `Terrorist` structures. `ReadOneTerrorist()` allocates memory on the heap for a `TerroristStruct`, initializes the struct using the data file, and then returns a pointer to the newly allocated and initialized `TerroristStruct`. Each `TerroristStruct` should have a pointer to a `CrimeStruct`, which allows you to dynamically allocate an array of `CrimeStruct`s of any size as you read in the file. Note the use of the `StringToReal` and `StringToInteger` functions from `strlib.h`. `ReadLine()` reads every line in as a string, so you must explicitly convert back to the orginal data type.

```
#define MAX_TERRORISTS 10
#define DATAFILE “terrorists.dat”

typedef struct {
   string description;
   string location;
   int year;
} CrimeStruct;

typedef struct {
   string name;
   string alias;
   double height;
   double weight;
   int numCrimes;
   CrimeStruct *crimes;
} TerroristStruct;

void InitTerroristArray(TerroristStruct *terrorists[], int size);
TerroristStruct *ReadOneTerrorist(FILE *infile);

void main(void)
{
   TerroristStruct *terrorists[MAX_TERRORISTS]
   FILE *infile;
   int numTerrorists;
   int i;

   infile = fopen(DATAFILE, “r”);
   if (infile==NULL) {
         Error(“Cannot open specified data file: %s...\n”, DATAFILE);
   }

   numTerrorists = StringToInteger(ReadLine(infile));
   ReadLine(infile);          // swallow up the blank line
   if (numTerrorists>MAX_TERRORISTS) {
         Error(“Too many terrorists.\n”);
   }

   InitTerroristArray(terrorists[], MAX_TERRORISTS);

   for(i=0; i<numTerrorists; i++) {
         terrorists[i]=ReadOneTerrorist(infile);
         ReadLine(infile);                // swallow up the blank line
   }
}
```

```
/****************************
 * Function InitTerroristArray
 ****************************
 * InitTerroristArray initializes the
 * provided array of pointers to NULL.
 * This is just to make sure we don’t
 * try to dereference uninitialized memory later on.
 ****************************/
void InitTerroristArray(TerroristStruct *terrorists[], int size)
{
   int i;
   for (I=0; i<size; i++) {
         terrorists[i]=NULL;
   }
}


/****************************
 * Function: ReadOneTerrorist
 ****************************
 * ReadOneTerrorist dynamically allocates a
 * TerroristStruct on the heap, and initializes
 * it using the information read in from the
 * supplied file.  ReadOneTerrorist assumes that
 * infile has been opened for reading and that the
 * file pointer is pointing at the first line of a
 * terrorist description.  It returns a pointer to
 * the newly allocated and initialized structure.
 ****************************/
TerroristStruct* ReadOneTerrorist(FILE *infile)
{
   TerroristStruct *aTerrorist;
   int i;

   aTerrorist = GetBlock(sizeof(TerroristStruct));
   aTerrorist->name = ReadLine(infile);
   aTerrorist->alias = ReadLine(infile);
   aTerrorist->height = StringToReal(ReadLine(infile));
   aTerrorist->weight = StringToReal(ReadLine(infile));
   aTerrorist->numCrimes = StringToInteger(ReadLine(infile));
   aTerrorist->crimes = NewArray(aTerrorist->numCrimes, CrimeStruct);
   for (i=0; i<aTerrorist->numCrimes; i++) {
         aTerrorist->crimes[i].description = ReadLine(infile);
         aTerrorist->crimes[i].location = ReadLine(infile);
         aTerrorist->crimes[i].year = StringToInteger(ReadLine(infile));
   }
   return aTerrorist;
}
```