Our implementation will be similar to the one in the stack2.c file of Section 19.2. The linked list will consist of nodes, represented by the following structure:

```
struct node {
  Item data;
  struct node *next;
};
```

The type of the data member is now Item rather than int, but the structure is otherwise the same as before.

The stack_type structure will contain a pointer to the first node in the list:

```
struct stack_type {
  struct node *top;
};
```

At first glance, the stack_type structure seems superfluous; we could just define Stack to be struct node * and let a Stack value be a pointer to the first node in the list. However, we still need the stack_type structure so that the interface to the stack remains unchanged. (If we did away with it, any function that modified the stack would need a Stack * parameter instead of a Stack parameter.) Moreover, having the stack_type structure will make it easier to change the implementation in the future, should we decide to store additional information. For example, if we later decide that the stack_type structure should contain a count of how many items are currently stored in the stack, we can easily add a member to the stack_type structure to store this information.

We won't need to make any changes to the stackADT.h header. (We'll use this header file, not stackADT2.h.) We can also use the original stackclient.c file for testing. All the changes will be in the stackADT.c file. Here's the new version:

stackADT3.c

```
#include <stdio.h>
#include <stdlib.h>
#include "stackADT.h"

struct node {
  Item data;
  struct node *next;
};

struct stack_type {
  struct node *top;
};

static void terminate(const char *message)
{
  printf("%s\n", message);
  exit(EXIT_FAILURE);
}
```