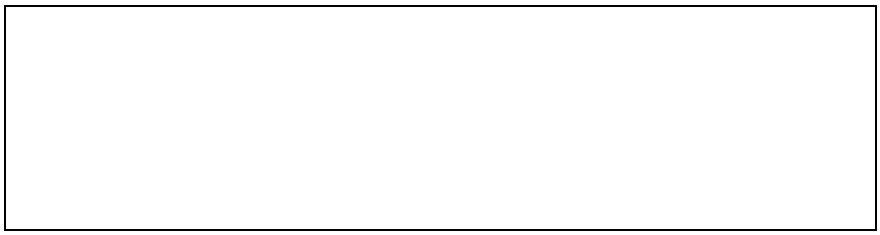
Worksheet 17: Linked List Introduction, List Stack

Group 11

Danny Mejia
Tatyana Vlaskin
Katherin Jensen

In Preparation: Read Chapter 6 to learn more about the Stack data type.

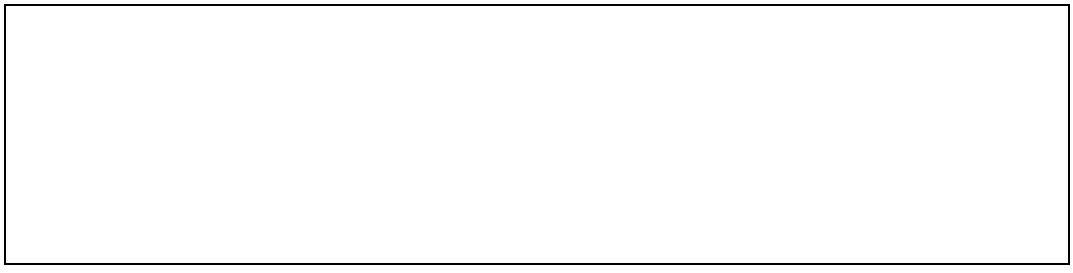
In Worksheet 14, you built a Stack using a dynamic array as the underlying container. A weakness of the Dynamic Array is that elements are stored in a contiguous block. As a consequence, when a new element is inserted into the middle of the collection, all the adjacent elements must be moved in order to make space for the new value.



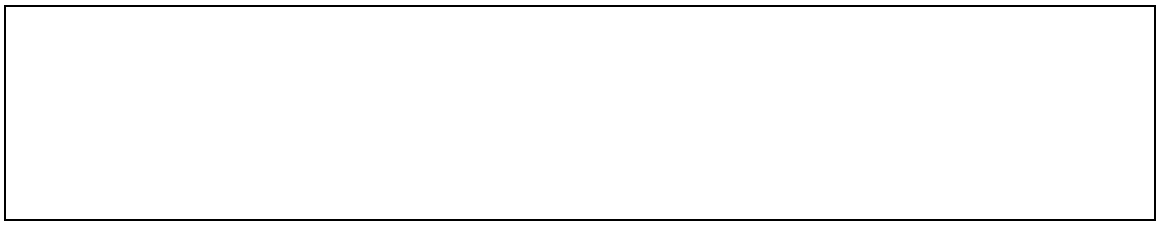
An alternative approach is to use the idea of a *Linked List*. In a linked list each value is stored in a separate block of memory, termed a *link*. In addition to a value, each link contains a reference to the next link in sequence. As a data structure, a link can be described as shown at right.

```
struct link {
   TYPE value;
   struct link * next;
};
```

We can visualize collection formed out of links as follows. A data field named firstLink will hold the first link in the collection. Each link refers to the next. The final link will have a **null** value in the next field:



The simplest data structure to create using links is a Stack. When a new element is pushed on the stack a new link will be created and placed at the front of the chain.



To remove a link the variable **firstLink** is simply changed to point to the next element in the chain. The space for the Link must then be freed.

The following is the beginning of an implementation of a **LinkedListStack** based on these ideas. Complete the implementation. Each operation should have constant time performance. Use an assertion to ensure that when a top or pop is performed the stack has at least one element. When you pop a value from the stack, make sure you free the link field.

```
struct link {
   TYPE value;
   struct link * next;
};

struct linkedListStack {
   struct link *firstLink;
}

void linkedListStackInit (struct linkedListStack * s)
            { s->firstLink = 0; }

void linkedListStackFree (struct linkedListStack *s)
      { while (! linkedListStackIsEmpty(s)) linkedListStackPop(s); }


//ADDING THE NODE TO THE LIST
void linkedListStackPush (struct linkedListStack *s, TYPE d) {
   struct link * newLink = (struct link *) malloc(sizeof(struct link));
   assert (newLink != 0);
   newLink->value = d;  // SET THE LINK VALUE EQUAL TO D
   newLink->next=s->firstLink; //REASSINGS THE POINTER
   s->firstLink = newLink;
}

TYPE linkedListStackTop (struct linkedListStack *s) {

   assert (!linkedListStackIsEmpty(s));
   return s->firstLink->value; // DISPLAY THE VALUE OF THE FIST LINK

}
//REMOVE THE 1ST NODE
void linkedListStackPop (struct linkedListStack *s) {
   struct link* name = s->fistLink
   assert (!linkedListStackIsEmpty(s));
   s->firstLink = name->next;
   free (name);
}
```

```
//CHECKS IF THE LINKED LIST IS EMPTY
int linkedListStackIsEmpty (struct linkedListStack *s) {
        return s->fistLink ==0;
}
```

REFFERENCE: https://gist.github.com/BAKERDAR/7029931