Worksheet 17b: Linked List Bag (version 1)

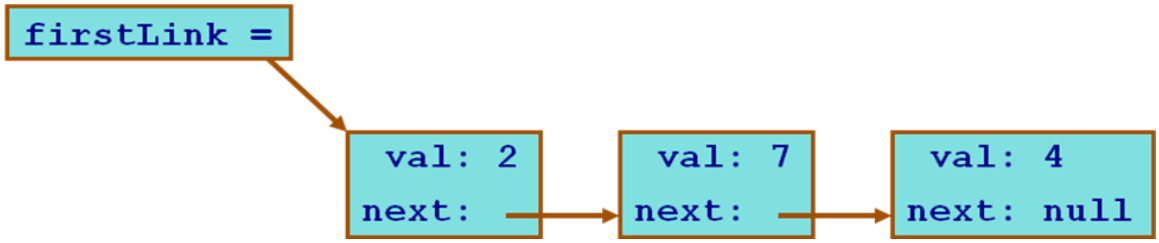
In Preparation: Read Chapter 6 to learn more about the Stack data type.

In Worksheet 17 you built a Stack using a Linked list.

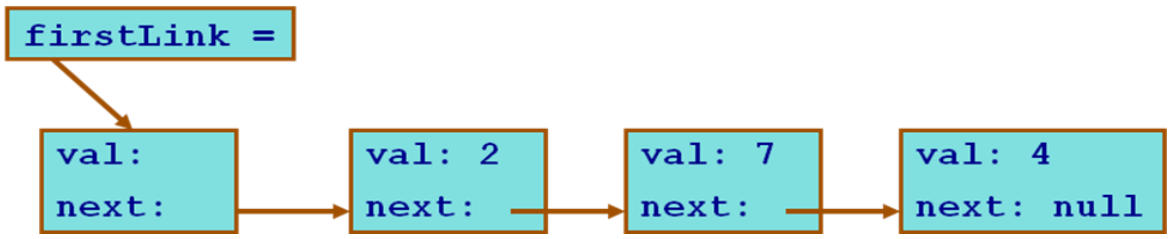
Remember that in a linked list each value is stored in a separate block of memory, termed a *link*. In addition to a value, each link contains a reference to the next link in sequence. As a data structure, a link can be described as shown at right.

```
struct link {
   EleType 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:



In worksheet 17 you build a stack using this idea of a linked list. This worksheet you will build a bag. Remember the bag operations are add, test, remove and size. We are going to ignore size for now (will come back to that in a later worksheet). Add is just the same as push for the stack.



The Contains test is also pretty simple. Loop over each link, testing each value. If you find a link containing the value you are seeking, return true. Otherwise, when you get to the last link, return false.

The only tricky operation is remove, since this requires changing the link on the value BEFORE the item you want to remove. There are a couple different ways to approach this. One way is to use double links. We will explore this in a later worksheet. The other approach is to use a *previous pointer*. As you loop over the elements in a list, the

previous pointer keeps a reference to the last link. It is zero initially. The loop for the previous pointer is shown in the remove method:

Using these ideas finish the implementation of the BAG

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

struct linkedListBag {
   struct link *firstLink;
}

void linkedListBagInit (struct linkedListBag * s)
            { s->firstLink = 0; }

struct link * _newLink (EleType v, struct link * n) { /* internal */
   struct link * lnk = (struct link *) malloc(sizeof(struct link));
   assert (lnk != 0);
   lnk->value = v;
   lnk->next = n;
   return lnk;
}

void linkedListBagAdd (struct linkedListBag *s, EleType d) {
   s->firstLink = _newLink (d, s->firstLink);
}

int linkedListBagContains (struct linkedListBag *s, EleType e) {









}

void linkedListBagRemove (struct linkedListBag *lst, EleType e) {
  struct link * prevLink =                              ;
  struct link * current =                               ;
  for (current =            ; current != 0;  current = current->next) {
      if (EQ(current->value, e)) {



      }




   }
}
```