Jordan Bayles
Cameron Evensen
Marshal Horn
Robert Plascencia

JCMR, Worksheet 22

1 Code implementation

Code 1: Worksheet #22 - Code implementation

```
int linkedListContains( struct linkedList *lst, TYPE e )
{
assert( lst != NULL );
struct dlink *tmp_lnk = lst->frontSentinel;

while( tmp_lnk->next != NULL )
    {
    if( tmp_lnk->value == e )
        {
        return( 1 );
        }
    tmp_lnk = tmp_lnk->next;
    }
return( 0 );

}

int linkedListRemove( struct linkedList *lst, TYPE e )
{
assert( lst != NULL );
struct dlink *tmp_lnk = lst->frontSentinel;

while( tmp_lnk->next != NULL )
    {
    /* We have found the link to remove */
    if( tmp_lnk->value == e )
        {
        _removeLink( lst, tmp_lnk );
        }
    tmp_lnk = tmp_lnk->next;
    }
return( 0 );

}
```

2 Question responses

2.1 What were the algorithmic complexities of the methods addLink and removeLink that you wrote back in Chapter Q?

Given that the search phrase "Chapter Q" gives exactly one result for the class documentation (this worksheet), I will assume that is some sort of type or stale reference. However, based upon the methods written in this worksheet the answer is much clearer. The `_addLink` function accepts a pointer to the list and the link to add before, same for the `_removeLink` function.

2.2 Given your answer to the previous question, what are the algorithmic complexities of the three principle Bag operations?

Because the `_addLink` and `_removeLink` functions are constant time (the nearby link is passed in), the algorithmic complexities of the add, remove and contains Bag operations can be found in their respective linkedList* functions:

- `linkedListAdd` is constant time (front of linkedList), or $\mathcal{O}(1)$.
- `linkedListContains` is linear time, as it must iterate through the linked list in order to find the element to return its value. Worst case is n operations, so the contains operation is $\mathcal{O}(n)$.
- Although the actual remove step is constant time, because `linkedListRemove` must first iterate through the list to find the element to remove (similar to the contains function), `linkedListRemove` is also $\mathcal{O}(n)$.