**Project**

ADT Bag - implementation on a doubly linked list on an array

1. **ADT Bag**

**Domain:**

B = {b | b is a Bag with elements of type TElem}

**Interface (set of operations):**

init(b)

pre : true

post: b ϵ B, b is an empty Bag

add(b, e)

pre: b ϵ B, e ϵ TElem

post: b’ ϵ B, b’ = b U {e} (TElem e is added to the Bag)

remove(b, e)

pre: b ϵ B, e ϵ Telem

post: b’ ϵ B, b’ = b \ {e} (one ocurrence of e was removed from the Bag). If e is not in b, b is not changed.

search(b, e)

pre: b ϵ B, e ϵ TElem

post: search <- true, if e ϵ B; false, otherwise

size(b)

pre: b ϵ B

post: size <- the number of elements from b

destroy(b)

pre: b ϵ B

post: b was destroyed

iterator(b, it)

pre: b ϵ B

post: it ϵ I, it is an iterator over b

**Representation:**

Bag:

elems: DLLANode[]

head: Integer

tail: Integer

firstEmpty: Integer

max: Integer

count: Integer

DLLANode:

info: Telem

next: Integer

prev: Integer

**1.1 BagIterator**

**Domain**: I = {i | i is an iterator over b ϵ B }

**Interface(set of operations):**

init(it, b)

pre: b ϵ B

post: it ϵ I, it is an iterator over b

valid(it)

pre: it ϵ I

post: valid <- true, if the current element from it is a valid one; false, otherwise

next(it)

pre: it ϵ I, valid(i)

post: it’ ϵ I , the current element from it’ refers to the next element from the bag b.

getCurrent(it, e)

pre: it ϵ I, valid(it)

post: e ϵ TElem, e is the current element from it

**Representation:**

BagIterator:

b: Bag

current: Integer

1. **Tests**

void Tests::testBag()

{

Bag b;

assert(b.size() == 0);

b.add("elem1");

b.add("elem2");

b.add("elem3");

b.add("elem4");

assert(b.size() == 4);

b.remove("elem4");

b.remove("elem1");

assert(b.size() == 2);

assert(b.search("elem2") == true);

assert(b.search("elem4") == false);

}

1. **Problem statement:**

A dog adoption center has to keep track of all the dogs that it has, given that new dogs are being brought in every day, and some are being adopted. To make the adoption process easier, the staff needs access to the information whether they have a dog of a certain breed in the center, and whether there are any dogs in the center up for adoption.

This problem can be solved using a ADT Bag, because there are going to be duplicates(multiple dogs of the same breed) and there is no need for ordering or positions. The operations needed are: add, search, remove, and size. These are exactly the operations of the ADT Bag, and no extra time or memory is spent on extra features(sorting, positions) that wouldn’t get used.