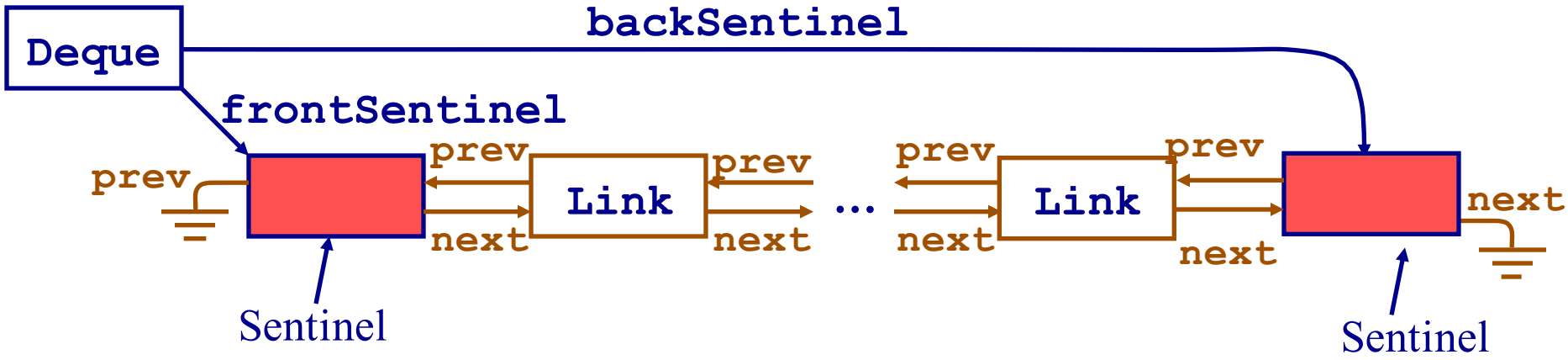
CS 261: Data Structures

Double Linked List Bag

Deque

```
struct dlink{
   TYPE value;
   struct dlink *next;
   struct dlink *prev;
};
```

```
struct listDeque{
   int size;
   struct dlink * frontSentinel;
   struct dlink * backSentinel;
};
```



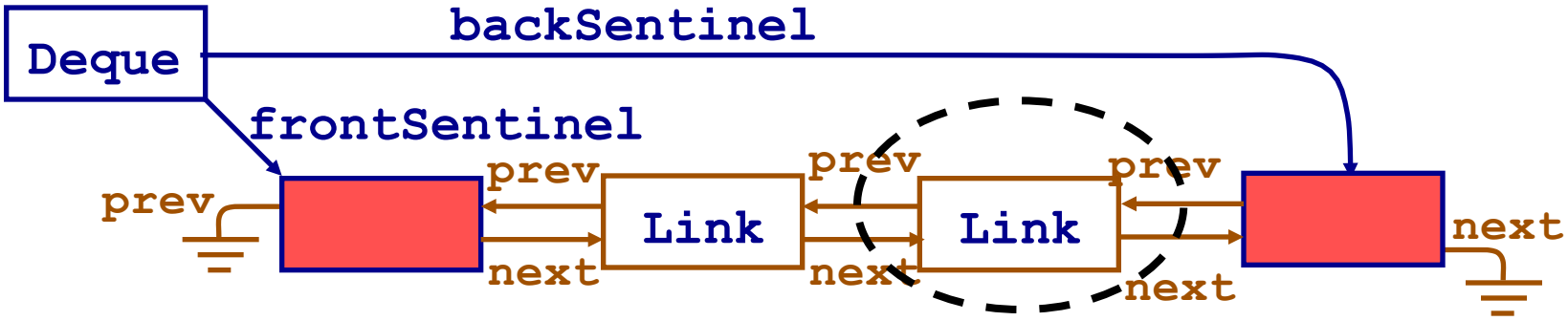
Remove from Front or Back

```
void removeFrontDeque (struct listDeque *dq)
{
      _removeDeque(dq, dq->frontSentinel->next);
}
```

```
void removeBackDeque (struct listDeque *dq)
{
      _removeDeque(dq, dq->backSentinel->prev);
}
```

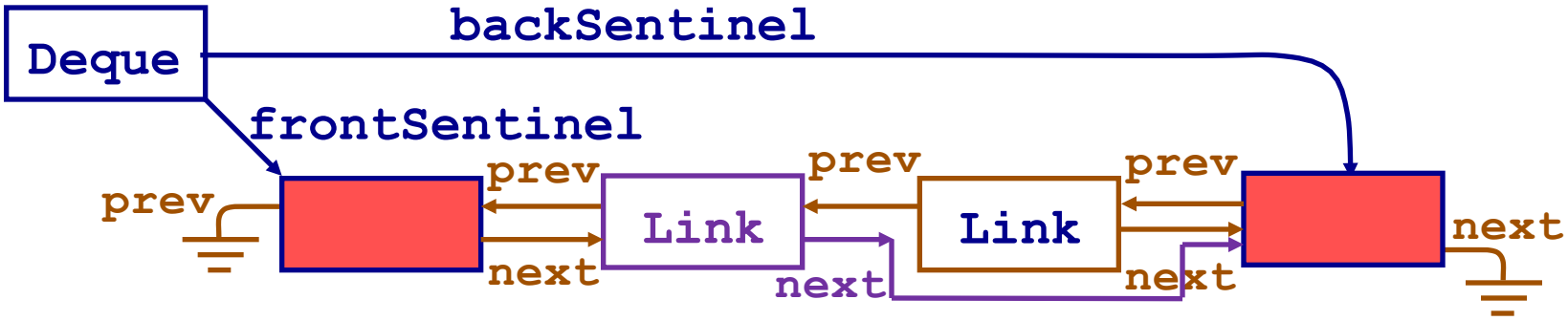
Remove Double Link from Deque

```
void _removeDeque (struct linkedList *dq,
                   struct dlink *lnk)
{
      assert(!isEmptyDeque(dq));




}
```



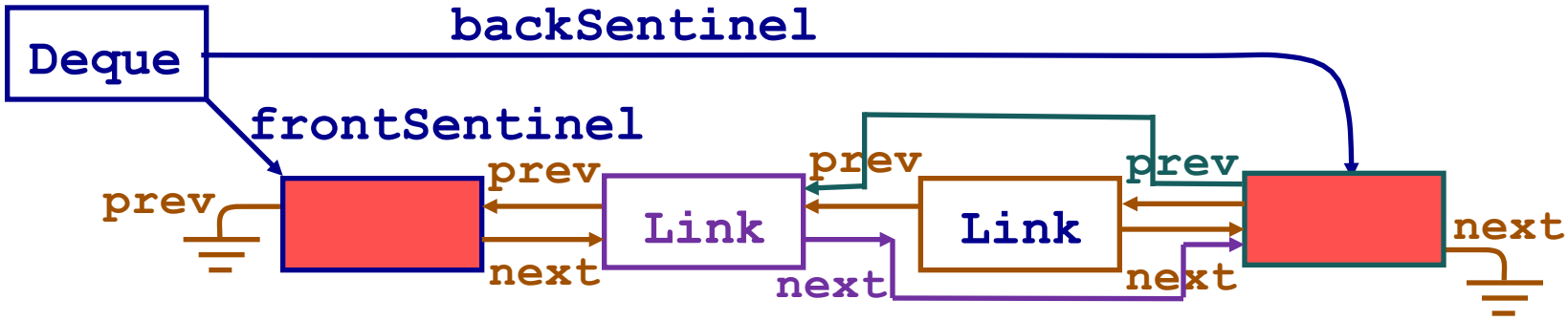
Remove Double Link from Deque

```
void _removeDeque (struct linkedList *dq,
                   struct dlink *lnk)
{
        assert(!isEmptyDeque(dq));
        lnk->prev->next = lnk->next;



}
```



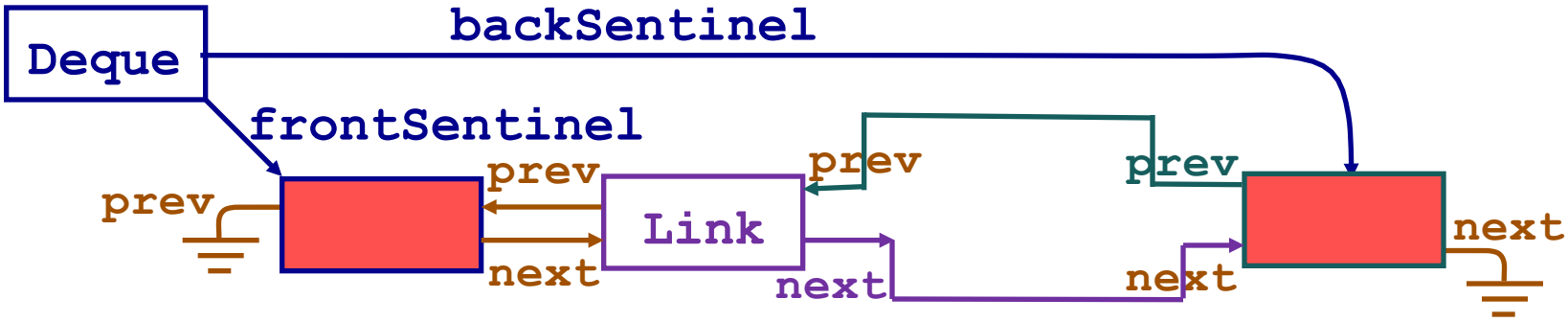
Remove Double Link from Deque

```
void _removeDeque (struct linkedList *dq,
                   struct dlink *lnk)
{
        assert(!isEmptyDeque(dq));
        lnk->prev->next = lnk->next;
        lnk->next->prev = lnk->prev;



}
```



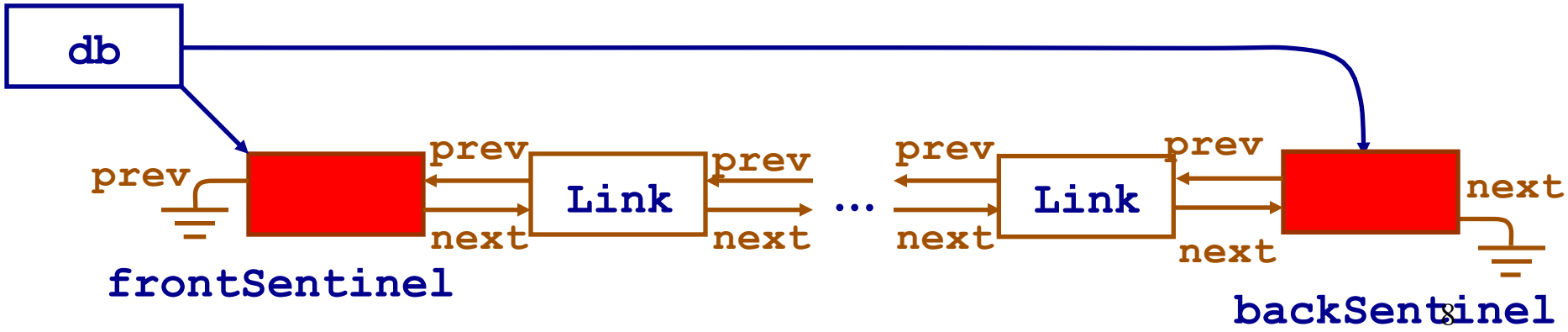
Remove Double Link from Deque

```
void _removeDeque (struct linkedList *dq,
                   struct dlink *lnk)
{
        assert(!isEmptyDeque(dq));
        lnk->prev->next = lnk->next;
        lnk->next->prev = lnk->prev;
        free(lnk);
        dq->size--;
}
```



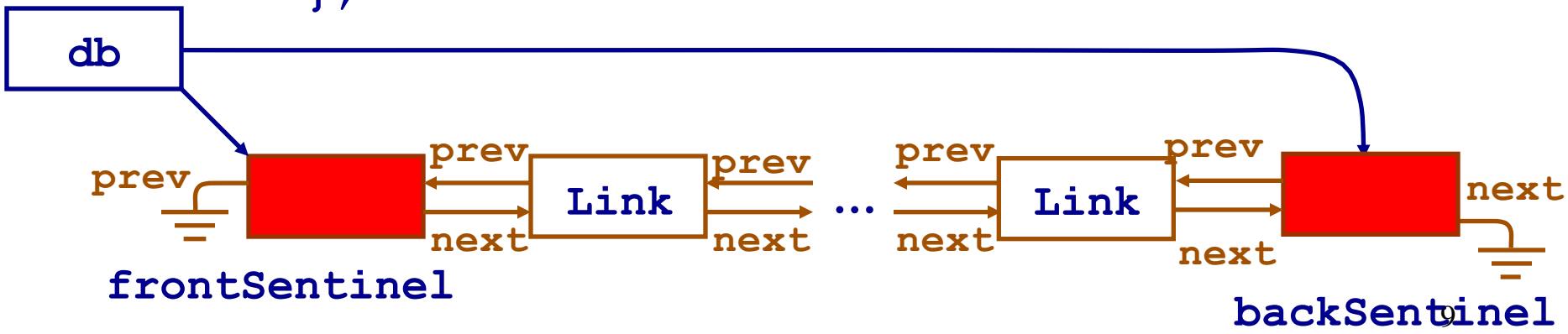
D-Bag

Arbitrary ordering of elements in the collection



D-Bag Structure

```
struct dlink {
    TYPE value;
    struct dlink * next;
    struct dlink * prev;
};
struct listDBag {
    int size;
    struct dlink * frontSentinel;
    struct dlink * backSentinel;
};
```

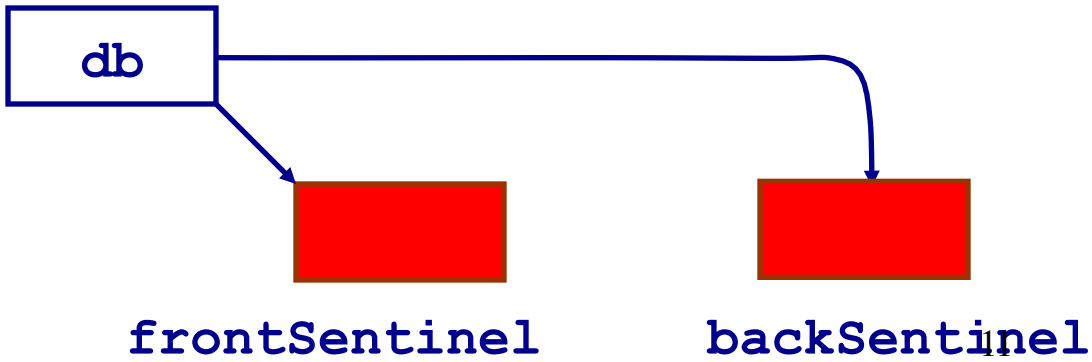


Bag Interface

```
void initBag();     /* Initialize the bag*/

int isEmptyBag();   /* Check if the bag is empty*/

void addBag();      /* Add new data element */

void removeBag();   /* Remove a given data element */

void containsBag();/* Check if a value is in the bag*/
```

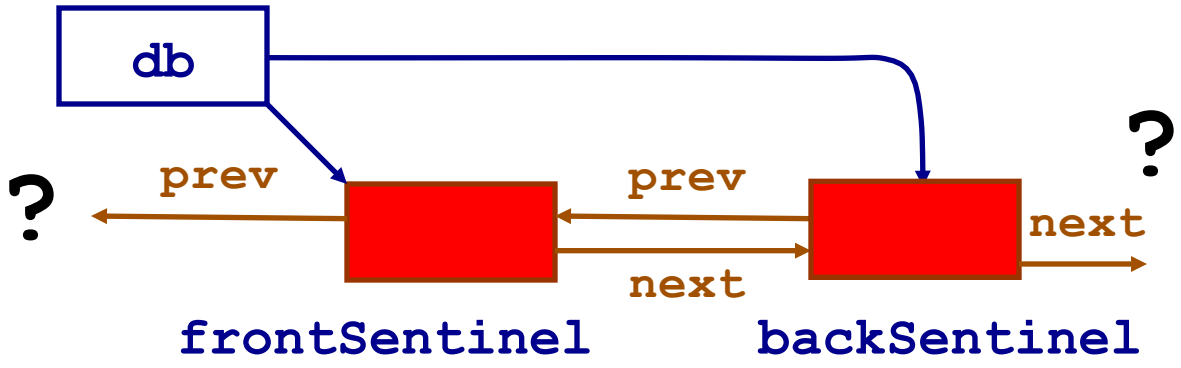
initDBag

```
void initDBag (struct listDBag *db) {
   assert(db);
   db->frontSentinel = (struct dlink *)
                       malloc(sizeof(struct dlink));
   assert(db->frontSentinel != 0);
   db->backSentinel = (struct dlink *)
                       malloc(sizeof(struct dlink));
   assert(db->backSentinel != 0);
   ...

}
```



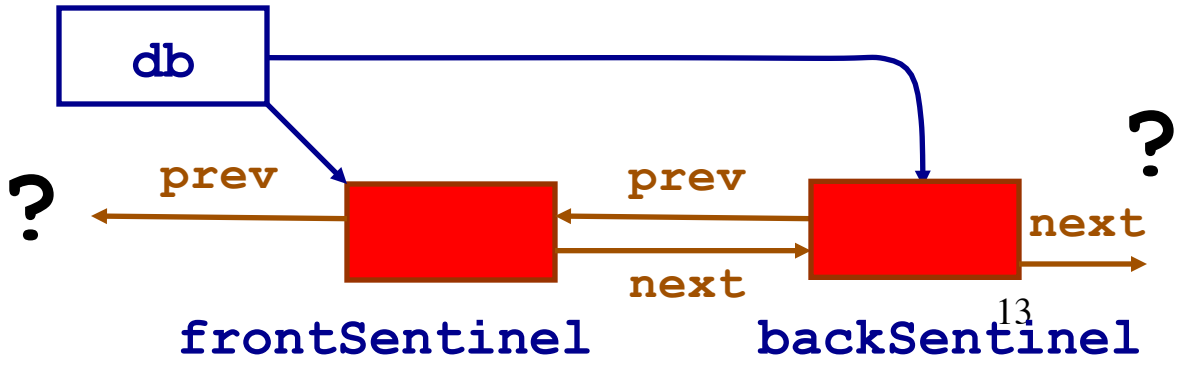
initDBag

```
void initDBag (struct listDBag *db) {
   assert(db);
   db->frontSentinel = (struct dlink *)
                       malloc(sizeof(struct dlink));
   assert(db->frontSentinel != 0);
   db->backSentinel = (struct dlink *)
                       malloc(sizeof(struct dlink));
   assert(db->backSentinel != 0);
   db->frontSentinel->next = db->backSentinel;
   db->backSentinel->prev = db->frontSentinal;
   db->size = 0;
}
```



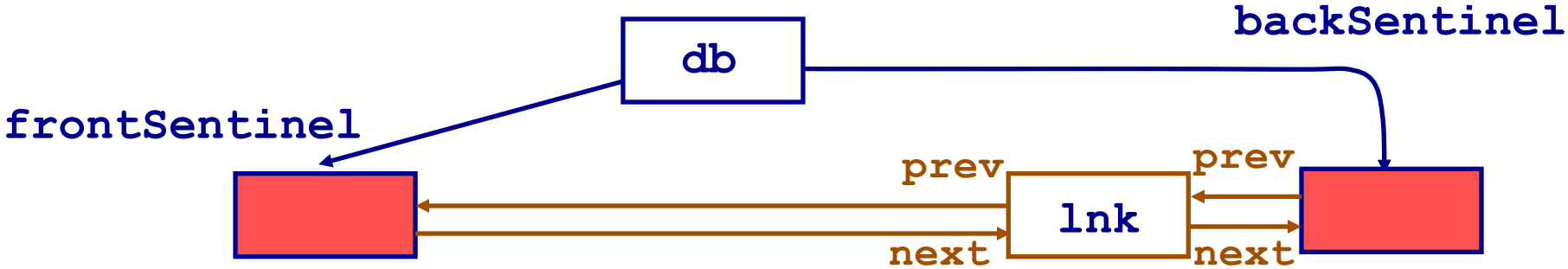
isEmpty

```
void isEmptyDBag (struct listDBag *db){

  assert(db);

  return db->size == 0;

}
```



AddDBag

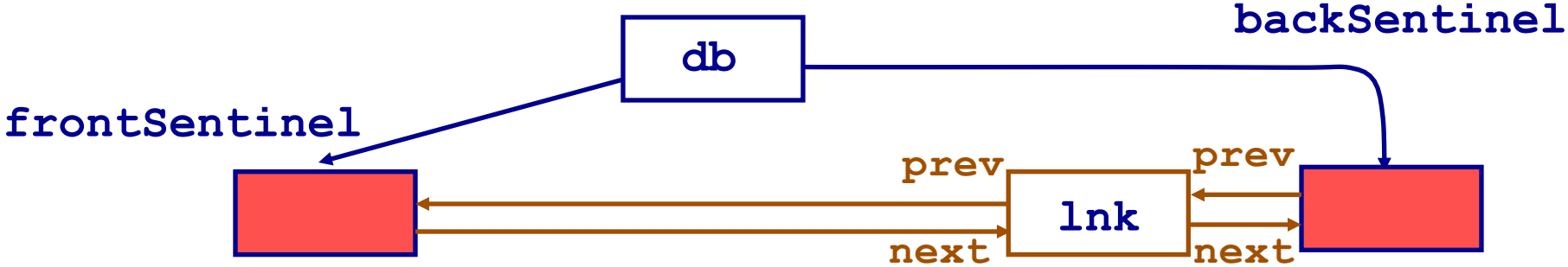
```
void addDBag (struct listDBag *db, TYPE e) {
    assert(db);
    _addDList(db, db->frontSentinel->next, e);
}
```



AddDList

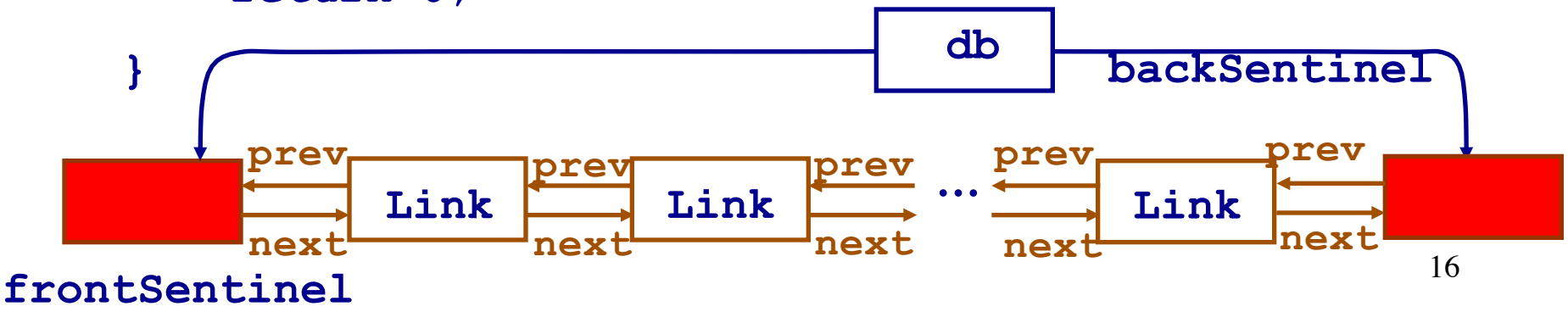
```
void _addDList (struct listDBag *db, struct dlink *lnk, TYPE e) {

}
```

VOLUNTEERS !



Contains Bag

```
int containsDBag (struct listDBag *db, TYPE e) {
    struct dlink *current;
    assert(!isEmptyBag(db));
    current = db->frontSentinel->next;
    while(current != db->backSentinel){
        if(current->value == e) return 1;
        current = current->next;
    }
    return 0;
}
```



Remove Bag

```
void removeDBag (struct listDBag *db, TYPE e) {
```

VOLUNTEERS !

```
}
```