Worksheet 19 ANSWER: Linked List Deque

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

struct linkedList {
   int size;
   struct dlink * frontSentinel;
   struct dlink * backSentinel;
};

             /* these functions are written for you */
void LinkedListInit (struct linkedList *q) {
   q->frontSentinel = malloc(sizeof(struct dlink));
   assert(q->frontSentinel != 0);
   q->backSentinel = malloc(sizeof(struct dlink));
   assert(q->backSentinel);
   q->frontSentinel->next = q->backSentinel;
   q->backSentinel->prev = q->frontSentinal;
   q->size = 0;
}

void linkedListFree (struct linkedList *q) {
   while (q->size > 0)
      linkedListRemoveFront(q);
   free (q->frontSentinel);
   free (q->backSsentinel);
   q->frontSentinal = q->backSentinal = null;
}

void LinkedListAddFront (struct linkedList *q, TYPE e)
   { _addBefore(q, q->frontSentinel->next, e); }

void LinkedListAddback (struct linkedList *q, TYPE e)
   { _addBefore(q, q->backSentinel, e); }

void linkedListRemoveFront (struct linkedList *q) {
   assert(! linkedListIsEmpty(q));
   _removeLink (q, q->frontSentinel->next);
}

void LinkedListRemoveBack (struct linkedList *q) {
   assert(! linkedListIsEmpty(q));
   _removeLink (q, q->backSentinel->prev);
}

int LinkedListIsEmpty (struct linkedList *q) {
   return q->size == 0;
}


/* write addLink and removeLink. Make sure they update the size field correctly */

void _addBefore (struct linkedList *q, struct dlink *lnk, TYPE e) {
    struct dlink * newlink = malloc(sizeof(struct dlink));
    assert(newlink != 0);
     newlink->value = e;
     newlink->prev = lnk->prev;
     newlink->next = lnk;
     lnk->prev->next = newlink;
     lnk->prev = newlink;
     q->size++;
}

void _removeLink (struct linkedList *q, struct dlink *lnk) {
    lnk->prev->next = lnk->next;
    lnk->next->prev = lnk->prev;
    free(lnk);
    q->size--;
}

TYPE LinkedListFront (struct linkedList *q) {
     assert(! LinkedListIsEmpty(q));
     return q->frontSentinel->next->value;
}

TYPE LinkedListBack (struct linkedList *q) {
     assert(! LinkedListIsEmpty(q));
     return q->backSentinel->prev->value;
}
```