```c
#include "deque.h"

/* storage allocated here */
struct node hdr;

/* ADT interface functions */
/* THink of hdr as if it is on top
of all member nodes of deque.
Left and right of header is not symmetric to
those of member nodes */
void init() {
    // unused
    hdr.data = 0;
    hdr.nextL = hdr.nextR = NULL;
}

void joinL(int d) {
    printf("Going to join %d on left\n", d);
    struct node *new = malloc(sizeof(struct node));
    assert(new!=NULL); // Stop if problem

    new->data = d;

    if (hdr.nextL == NULL) {
        assert(hdr.nextR == NULL);
        hdr.nextL = hdr.nextR = new;
        new->nextL = new->nextR = NULL;
        printf("Joined %d on left\n", d);
        return;
    }

    assert(hdr.nextR != NULL);
    assert(hdr.nextL->nextL == NULL);
    hdr.nextL->nextL = new;
    new->nextR = hdr.nextL;
    new->nextL = NULL;
    hdr.nextL = new;
    printf("Joined %d on left\n", d);
}

void joinR(int d) {
    return;
}

int leaveL() {
    // Unimplemented
    return 0;
}
int leaveR() {
```

```
50:     struct node *tmp;
51:     printf("Someone leaving from Right\n");
52:     assert(hdr.nextL != NULL && hdr.nextR != NULL);
53:     int d = hdr.nextR->data;
54:     tmp = hdr.nextR->nextL;
55:     if (tmp != NULL)
56:         tmp->nextR = NULL;
57:     free(hdr.nextR);
58:     hdr.nextR = tmp;
59:     printf("From right %d left\n", d);
60:     if (tmp == NULL)
61:         hdr.nextL = NULL;
62:     assert (tmp != NULL || hdr.nextL == NULL);
63:     return d;
64: }
65:
66: int size() {
67:     int i = 0;
68:     struct node *ptr = hdr.nextL;
69:
70:     while (ptr != NULL) {
71:         i++; ptr = ptr->nextR;
72:     }
73:     return i;
74: }
```