```
/*
ALEX FRIEDBERG

LINKED LIST .cpp
*/

#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include "linklist.h"

using namespace std;

linklist::linklist() {
    head = NULL;
}

/*
void linklist::removeName(string name){

LinkListEntry *curr;
LinkListEntry *prev;
prev = NULL;

for(curr = head; curr != NULL; curr = curr->next){
if(curr->name.compare(name) == 0){
if(prev == NULL){
//First Entry in list
head = curr->next;
delete curr;
}else{
//more then one entry in list
prev->next = curr->next;
delete curr;
}
return;
}
prev = curr;
}
cout << "ERROR: '" << name << "' not found in list" << endl;

}
*/

LINK_LIST_ENTRY_TYPE linklist::removeAtBegin() {
    LinkListEntry *prev;
    LinkListEntry *curr;
    LINK_LIST_ENTRY_TYPE returnValue;
    prev = NULL;

    for (curr = head; curr != NULL; curr = curr->next) {
        if (curr->next == NULL) {
            //Last Entry
            if (prev == NULL) {
                //First and only entry in list
                head = NULL;
            }
            else {
                prev->next = NULL;
            }
            returnValue = curr->data;
            delete curr;
            break;
        }
        prev = curr;
    }

    return returnValue;

}

LINK_LIST_ENTRY_TYPE linklist::removeAtEnd() {
    LinkListEntry *prev;
    LinkListEntry *curr;
    LINK_LIST_ENTRY_TYPE returnValue;

    if (head != NULL) {
        curr = head;
        head = head->next;

        returnValue = curr->data;
        delete curr;
    }
    return returnValue;
}


void linklist::addAtBegin(LINK_LIST_ENTRY_TYPE data) {

    LinkListEntry *n = new LinkListEntry;

    LinkListEntry *curr;
    LinkListEntry *prev;
    prev = NULL;

    for (curr = head; curr != NULL; curr = curr->next) {
```

```cpp
        prev = curr;
    }

    //  LinkListEntry *n = new linkListEntry;

    n->data = data;
    n->next = NULL;
    if (head != NULL) {
        prev->next = n;
    }
    else {
        head = n;
    }
}

void linklist::addAtEnd(LINK_LIST_ENTRY_TYPE data) {

    LinkListEntry *n = new LinkListEntry; //NEW NODE

    n->data = data;

    n->next = head;
    head = n;

}


void linklist::printList() {
    LinkListEntry *curr;

    for (curr = head; curr != NULL; curr = curr->next) {

        cout << curr->data;
        if (curr->next != NULL){
            cout << endl;
        }

    }
    cout << endl;


}

//Returns data at the beginning of the queue
LINK_LIST_ENTRY_TYPE linklist::peekAtBegin() {

    LinkListEntry *curr;
    LinkListEntry *prev;
    prev = NULL;
    //person dummyPerson;

    for (curr = head; curr != NULL; curr = curr->next) {
        prev = curr;
    }

    if (prev != NULL) {
        return prev->data;
    }

    return NULL;

}

LINK_LIST_ENTRY_TYPE linklist::peekAtEnd() {

    //person dummyPerson;

    if (head != NULL) {
        return head->data;
    }

    return NULL;

}

//Returns data value at the index of the list
LINK_LIST_ENTRY_TYPE linklist::getData(int index) {
    LinkListEntry *curr;
    int currIndex = 0;
    //person dummyPerson;

    for (curr = head; curr != NULL; curr = curr->next) {
        if (currIndex == index) {
            return curr->data;
        }
        currIndex++;
    }
    return NULL;
}

//Get number of entries in linklist
int linklist::countList() {
    LinkListEntry *curr;
    int currIndex = 0;

    for (curr = head; curr != NULL; curr = curr->next) {
        currIndex++;
    }
    return currIndex;
}
/*END OF FILE*/
```