CS 101 Spring 2018 Project 2 Due 3/7

For this assignment you will build a doubly linked list class. The name of your class is dList, and each entry in the linked list should store 2 different values, a key (integer) and a type (character). A struct (or class) defining the nodes of the linked list should appear outside of the dList class definition and should be named “node”. The class should support the following operations:

dList()	Default constructor. Should create an empty list.
dList(int[], char[],int)	Constructor where the first two parameters are arrays and the third is the length of the arrays. The constructor should initialize the list with the contents of the arrays.
void addFront(int, char)	Creates a new node at the front of the list.
void addBack(int, char)	Creates a new node at the back of the list.
node *search(int)	Searches the list for the occurrence of the int parameter in the list and returns a pointer to the node containing that key.
void find(char)	Outputs all keys that have the type equal to the character parameter. Should start at the front of the list and output the keys in list order.
void moveFront(node *)	Moves the node pointed to by the parameter to the front of the list.
void moveBack(node *)	Moves the node pointed to by the parameter to the back of the list.
void out(int, char = ‘f’)	Outputs the first k elements of the list, where k is the int parameter. The char parameter can be either ‘f’ or ‘b’. ‘f’ is the default value. If the char is ‘f’ the output starts at the front of the list. If the char is ‘b’, start at the end of the list and work backwards.
void sort()	Should perform a O(n lg n) time sorting algorithm on the items in the list. The list should be in increasing order based on the integer key after this operation.

You should be careful to manage your memory and implement a destructor, but it is not required to implement a copy constructor or assignment operator for the class.

You may use dummy header and tail nodes if you want.

Your code should be in a file named dList.cpp and the sample main below should compile and work without modification.

You are not allowed any include files in dList.cpp, notice that the main function will include iostream before the dList.cpp file.

```
#include <iostream>
using namespace std;
#include "dList.cpp"
#define SMALL 200
#define MAX 100000
#define ROUNDS 100
int main(){
    int i, x[MAX];
    char ch[MAX];

    for(i=0;i<SMALL;i++) {x[i] = 2*SMALL-i; ch[i] = 'a'+ (i%26);}
    dList A(x,ch,SMALL), B;
    A.out(10);
    node *tmp = A.search(2*SMALL-8);
    A.moveFront(tmp);
    A.out(10);
    A.moveBack(tmp);
    A.out(10);
    A.find('b');
    A.sort();
    A.out(10);
    A.out(10,'b');
    A.addBack(500,'d');
    A.addFront(501,'z');
    A.out(10);
    A.out(10,'b');
    B.addFront(1,'a');
    B.addBack(2,'b');
    B.out(2);

    for(int j=0; j<ROUNDS; j++){
        cout << endl << "round " << j << endl;
        for(i=0;i<MAX;i++) {x[i] = 2*MAX-i; ch[i] = 'a'+ (i%26);}
        dList A(x,ch,MAX);
        node *tmp = A.search(2*MAX-8);
        A.moveFront(tmp);
        A.moveBack(tmp);
        A.sort();
        A.out(10);
        A.out(10,'b');
    }
}
```

Example output:

You will notice a unix command named ulimit below. Ulimit is used to control the amount of memory that a process can use. I will use the ulimit command to determine if your code has a memory leak. As you see below, the solution code runs with a ulimit of 17000 for 100 rounds and would continue indefinitely. It is ok if your program needs a slightly higher ulimit, say 20000, but it should not have any memory leaks. The execution below takes just under 1 second per round on the server.

NOTE: You will have to raise the ulimit to be able to run other programs, like editing or compiling!

```
bdixon@cs-intro:~> ulimit -Sv 100000
bdixon@cs-intro:~> g++ dListSampleMain.cpp -O3 -o Project2
bdixon@cs-intro:~> ulimit -Sv 17000
bdixon@cs-intro:~> ./Project2
400 a  399 b  398 c  397 d  396 e  395 f  394 g  393 h  392 i  391 j
392 i  400 a  399 b  398 c  397 d  396 e  395 f  394 g  393 h  391 j
400 a  399 b  398 c  397 d  396 e  395 f  394 g  393 h  391 j  390 k
399 b  373 b  347 b  321 b  295 b  269 b  243 b  217 b
201 r  202 q  203 p  204 o  205 n  206 m  207 l  208 k  209 j  210 i
400 a  399 b  398 c  397 d  396 e  395 f  394 g  393 h  392 i  391 j
501 z  201 r  202 q  203 p  204 o  205 n  206 m  207 l  208 k  209 j
500 d  400 a  399 b  398 c  397 d  396 e  395 f  394 g  393 h  392 i
1 a  2 b

round 0
100001 d  100002 c  100003 b  100004 a  100005 z  100006 y  100007 x
100008 w  100009 v  100010 u
200000 a  199999 b  199998 c  199997 d  199996 e  199995 f  199994 g
199993 h  199992 i  199991 j

round 1
100001 d  100002 c  100003 b  100004 a  100005 z  100006 y  100007 x
100008 w  100009 v  100010 u
200000 a  199999 b  199998 c  199997 d  199996 e  199995 f  199994 g
199993 h  199992 i  199991 j

round 2
100001 d  100002 c  100003 b  100004 a  100005 z  100006 y  100007 x
100008 w  100009 v  100010 u
200000 a  199999 b  199998 c  199997 d  199996 e  199995 f  199994 g
199993 h  199992 i  199991 j

…

round 99
100001 d  100002 c  100003 b  100004 a  100005 z  100006 y  100007 x
100008 w  100009 v  100010 u
200000 a  199999 b  199998 c  199997 d  199996 e  199995 f  199994 g
199993 h  199992 i  199991 j

bdixon@cs-intro:~>
```