Ahmad Mohammad

c1001633 / asm6t

proj2

		deduction
TURNIN TIME	turned in on time.	0 %
SOURCE CODE SEARCH RESULTS	MISSING: None. FOUND: INT_LINKED_LIST, intsllist.h, IntSLLNode, class IntSLList, ints.txt, printAll, insertInOrder, deleteVal, deleteAllVal, clearList	0

DEDUCTIONS	-10
FINAL GRADE	90

COMPILATION LOG

```
c1001633 MAKE PASS

MAKELOG OF STUDENT PROVIDED MAKEFILE:
g++ -c -g -O0 -std=c++11 -Wall intsllist.cpp -o intsllist.o -MMD -MF intsllist.d
g++ -c -g -O0 -std=c++11 -Wall proj2.cpp -o proj2.o -MMD -MF proj2.d
g++  -o proj2 intsllist.o proj2.o
intsllist.cpp: In member function ‘int IntSLList::deleteVal(int)’:
intsllist.cpp:111:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
```

c1001633 runlog begin:
Test 1:
Inputs:
4 a
5 a
2 a
4 a
5 d
4 d
2 a
3 a

2 D

Output:
(a)4->[4,0x55d89028ee60]
(a)5->[4,0x55d89028ee60]->[5,0x55d89028fe90]
(a)2->[2,0x55d89028feb0]->[4,0x55d89028ee60]->[5,0x55d89028fe90]
(a)4->[2,0x55d89028feb0]->[4,0x55d89028ee60]->[4,0x55d89028fed0]->[5,0x55d89028fe90]
(d)5->[2,0x55d89028feb0]->[4,0x55d89028ee60]->[4,0x55d89028fed0]
(d)4->[2,0x55d89028feb0]->[4,0x55d89028fed0]
(a)2->[2,0x55d89028feb0]->[2,0x55d89028ee60]->[4,0x55d89028fed0]
(a)3->[2,0x55d89028feb0]->[2,0x55d89028ee60]->[3,0x55d89028fe90]->[4,0x55d89028fed0]
(D)2->[3,0x55d89028fe90]->[4,0x55d89028fed0]
Clearing 3 0x55d89028fe90
Clearing 4 0x55d89028fed0

Test 2:
Inputs:
7 a
8 a
8 a
7 d
5 a
6 a
5 D

Output:
(a)7->[7,0x5589de13be60]
(a)8->[7,0x5589de13be60]->[8,0x5589de13ce90]
(a)8->[7,0x5589de13be60]->[8,0x5589de13ce90]->[8,0x5589de13ceb0]
(d)7->[0,0x5589de13be60]->[8,0x5589de13ce90]->[8,0x5589de13ceb0]
(a)5->[5,0x5589de13be60]
(a)6->[5,0x5589de13be60]->[6,0x5589de13ced0]
(D)5->[6,0x5589de13ced0]
Clearing 6 0x5589de13ced0

```
Manifest: proj2.cpp.lst
------------------------
------------------------
 Feb 10 19:20 proj2.cpp
```

```
// Ahmad Mohammad
// CSCI-3110-001
// Proj 2 Driver
// Due: 02/20/2022

// This file will serve as main and perform operations and instantiate objects from
// our class file (header) and our implementation file.


#include<iostream>
#include<fstream>
#include<string>

// including header file so we have access to our class
#include "intsllist.h"

using namespace std;


int main()
{
                // creating object of type class
    IntSLList chicken;
                // declaring and setting infile
                ifstream myin;
    myin.open("ints.txt");

                // setting variables
                int num;
                string letter;

                // pulling number to be operated on from file and making sure not empty
                while(myin>>num)
                {
                                // pulling letter right after number from file
                                myin >> letter;

                                // if letter is 'a' then insert num and cout info in for
mat
                                if(letter == "a")
                                {
                                                                chicken.insertInOrder(num)
;
                                                                cout <<'('<<letter<<')'<<n
um;
                                                                chicken.printAll();
                                }
                                // if letter is 'd' then delete num pulled from file tha
t
                                // occurs first in list  and cout info in format
                                else if(letter == "d")
                                {
                                                                cout <<'('<<letter<<')'<<n
um;
                                                                chicken.deleteVal(num);
                 chicken.printAll();

                                }
                                // if letter = 'D' then delete all numbers in list that
= num pulled from file
                                // and cout info in correct format
```

```
                                else if(letter == "D")
                                {
                                                                        cout <<'('<<letter<<')'<<n
um;
                                                                        chicken.deleteAllVal(num);
                    chicken.printAll();

                                }

                  }
                  // end prog
                  return 0;
   }

---------------------- intsllist.h: ------------------------
#ifndef INT_LINKED_LIST
#define INT_LINKED_LIST
#include<iostream>

class IntSLList {
        public:

                // Constructor
                IntSLList() {
                        head = nullptr;
                }

                //D Destructor
                ~IntSLList() {
                        clearList();
                }

                // prints the info content and address of each node in the list
                void printAll() const {
                        for (IntSLLNode *tmp = head; tmp != nullptr; tmp = tmp->next)
                                std::cout << "->[" << tmp->info << "," << tmp << "]";
                        std::cout << std::endl;
                }

                // Inserts node in order (see assignment specification for details)
                void insertInOrder(int);

                // Deletes an occurrence of argument (see assignment specification for
 details)
                int deleteVal(int);

                // Deletes all occurrences of argument (see assignment specification f
or details)
                void deleteAllVal(int el);

                // Clears the list (deallocates memory - see assignment specification
for details)
                void clearList();

        private:
                // Node stored in linked list
                struct IntSLLNode {
                        IntSLLNode(int el = 0) {
                                info = el;
                                next = nullptr;
                        }
```

```
                        int info;
                        IntSLLNode *next;
                };

                IntSLLNode *head;       // head of the list
};

#endif


\--------------------- intsllist.cpp: -----------------------
// Ahmad Mohammad
// CSCI-3110-001
// Project 2
// Due: 02/10/2022

// This is an implementation file for the IntSLList Class provided in the
// intsllist.h header file. I.e. every finction that was not defined in the class
// (inline) will be defined in this file.

#include "intsllist.h"
#include<iostream>


using namespace std;


// Function that adds every value aquired from infile to the singly linked list
// in ascending order. If duplicate values are aquired the new value will be added
// 'behind' the old value (new value will be pointing to old value).
void IntSLList::insertInOrder(int number)
{
        // declaration of pointers to be used in traversal/insertion
    IntSLLNode *newnode, *cur, *prev;

        // dynamically allocating memory to new node and giving value passed in param.

    newnode = new IntSLLNode(number);
    newnode -> info = number;

        // preparing cur and prev for traversal
        cur = head;
    prev = NULL;

        // checks if head is NULL and if so sets the newnode to be head (empty list)
    if(!head)
        {
            head = newnode;
            newnode->next = NULL;
        }
        // otherwise traverse until we find valie
    else
        {
                        // traversal loop which makes sure cur != NULL and stops when
we reach
                        // where we want to insert node.
            while(cur != NULL && cur->info <= number)
            {
                prev = cur;
                cur = cur->next;
            }
```

```
                        // sets head to equal new node if both cur and prev are NULL
                        // ie. while loop did not run
            if(prev == NULL)
            {
                head = newnode;
                newnode->next = cur;
            }
                        // otherwise set prev->next to newnode and newnode->next to cu
r
                        // in order to keep cur 2 nodes ahead of prev and not get seg
fault
            else
            {
                prev->next = newnode;
                newnode->next = cur;
            }
        }

}

// This function will remove the first instance of the number in arg from linked list
int IntSLList::deleteVal(int number)
{
        // declaration of pointers needed to traverse and find node to delete
    IntSLLNode *cur, *prev;

        cur = head;
        // checks if head = NULL and if so exits
    if(!head)
        return -1;

        // checks if head is node to be deleted if so deletes and returns cur which =
head
    if(head->info == number)
    {
        delete head;
                return cur->info;
                cur = head->next;
        head = cur;

    }
        //other wise node to be deleted is in list somewhere...
    else
    {

                // traversal loop to locate node to be deleted
        while(cur != NULL && cur->info != number)
        {
            prev = cur;
            cur = cur->next;
        }
                // makes sure cur data = number because if not we would get seg fault
                // trying to delete NULL node
        if(cur -> info == number)
        {
                        // makes sure all data from cur is wiped before delete and ret
urns value we
                        // were looking for
            prev-> next = cur->next;
                        cur-> next = NULL;
                        prev = cur;
```

This is the handle to the entire list, so before deleting this node, make sure to set a pointer to the next node

-5

These are never executed because they occur after the return

-3

prev->info doesn't exist anymore after deleting cur, because cur was equal to prev

-2

```
            delete cur;
                        return prev->info;
        }

         }
}

// This function will remove all instances of value in arg from linked list
void IntSLList::deleteAllVal(int number)
{
        // declaration of pointers used in traversal
    IntSLLNode *cur, *prev;

        // checks if head = null.. if so exits
    if(!head)
        return;
        // if head = number we want to delete we delete head and set head to cur
        // while head is = number we want gone
    if(head->info == number)
    {
        while(head->info == number)
        {
            cur = head->next;
            delete head;
            head = cur;
        }
    }
        // otherwise the number we want to delete is in the list somewhere...
    else
    {
                // set cur =  head for traversal
                cur = head;

                // while list is a list and cur's data != number, traverse the list
        while(cur != NULL && cur->info != number)
        {
            prev = cur;
            cur = cur->next;
        }
                // if cur -> info = number then enter while loop and delete nodes
                // until it is not equal to number
        if(cur != NULL && cur -> info == number)
        {
            while(cur->info == number)
            {
                prev->next = cur->next;
                delete cur;
                                prev = cur;
                                cur = cur->next;
            }
        }
    }
}
// This function will clear the whole list and for ever node deleting it will print
// clearing message
void IntSLList::clearList()
{
        // sets cur and makes sure head != NULL
        IntSLLNode *cur;
        while(head)
        {
```

```
                        // clearing message
                        cout << "Clearing " << head->info <<' '<<&head->info<< endl;
            //setting cur to head to delete bc we cant delete head
                        cur = head;
                        // setting head to next node in list
                        head = head -> next;
                        // deleting old head
                        delete cur;
        }

}
```