Sushil Khanal

Section 1

Project 6

**IMPLEMENTING A LINKED LIST WITH RECURSIVE FUNCTIONS**

**Introduction**

A linked list is a linear collection of data elements, in which linear order is not given by their physical placement in memory. Instead, each element points to the next.

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

In this project we are going to create a menu-driven and interactive program that can read and respond to commands entered by user, which should do a lot of operation manipulation, by implementing a linked list with recursive function.

**Data Structures**

The program contains .h file, .cpp file and main file. .h contains class, which has all function prototypes. .cpp file contains the code for the functions. The main file receives instructions from the user and implements the function accordingly.

**Functions**

The functions used in this program are:

List(), void insert(), void remove(), item a\_empty() these type of function are also known as public member functions.

Int length(), bool bEmpty(), bool isPresent(), item kth() these type of functions are also known as constant member functions.

friend std::ostream& operator<<(std::ostream& out\_s, const list& list1) is a friend function.

Node\* get\_node(const item entry, Node\* g) is a private function.

**The Main Program**

The menu function can be called anytime in the main program so that the it could perform certain operations. Using switch, the program interacts with the user. The program runs in a loop and the only way to break the loop is entering q.

**Code**

List.h

#include <iostream>

#ifndef LIST\_H

#define LIST\_H

using namespace std;

//class named list

class list {

public:

//defination of data type

typedef int item; // what goes in a list

//constructor

list() { first = NULL; }; //inline

//post-condition: The list was set as an empty list

//destructor

~list();

//modification member function

void insert(const item& entry);

//pre-condition: The list does not contain the entry value

//post-condition: data was entered in the appropriate position

void remove(item target);

//pre-condition: The list contains the target value

//post-condition: target was removed from the list

item a\_empty();

//post-condition: The list was reinitialized to be empty

//constant member function

item length();

//post-condition: The returned value was the length of the list

bool bEmpty() const { return first == NULL; } //inline

//post-condition: If the call list is empty, return true else false

bool isPresent(const item& target);

//post-condition: if the target is present, return true else false

item kth(item target);

//pre-condition: The list is not empty and 1<= target <= length().

//post-condition: The kth element in the list has returned

//friend function

friend std::ostream& operator<<(std::ostream& out\_s, const list& list1);

//post-condition: The contents of the list were written to the output stream

private:

//private data members

struct Node {

item data;

Node\* next;

};

Node\* first;

//Private Function

//allocates, initializes and returns the address of the new node

Node\* get\_node(const item entry, Node\* g);

};

#endif

List.cpp

#include <iostream>

#include <cassert>

#include <cstdlib>

#include "List.h"

using namespace std;

//destructor

list::~list()

{

Node\* temp;

while (first != NULL) {

temp = first;

first = first->next;

delete temp;

}

}

//this function returns the length of the call list

int list::length()

{

Node\* g;

int count = 0;

g = first;

while (g != NULL) {

++count;

g = g->next;

}

return count;

}

//this function empties the call list

int list::a\_empty()

{

Node\* temp;

while (first != NULL) {

temp = first;

first = first->next;

delete temp;

}

}

//this function returns true if value is present in the call list, else false

bool list::isPresent(const item& target)

{

Node\* prev;

prev = first;

while (prev != NULL && prev->data != target) {

prev = prev->next;

return (prev != NULL);

}

}

//this function returns the kth value

list::item list::kth(item target)

{

Node\* prev;

prev = first;

for (int i = 0; i < target; ++i) {

prev = prev->next;

}

return (prev->data);

}

//this function inserts the new element

void list::insert(const item& entry)

{

Node\* prev;

assert(!isPresent(entry));

if (first == NULL || entry < first->data) {

first = get\_node(entry, first);

}

else {

prev = first;

while (prev->next != NULL && prev->next->data < entry)

prev = prev->next;

prev->next = get\_node(entry, prev->next);

}

}

//this function removes the target value

void list::remove(item target)

{

Node\* prev;

assert(isPresent(target));

prev = first;

if (first->data == target) {

first = first->next;

delete prev;

}

else {

Node\* temp;

while (prev->next->data != target)

prev = prev->next;

temp = prev->next;

prev->next = temp->next;

delete temp;

}

}

//this function returns the address of the new node

list::Node\* list::get\_node(const item entry, Node\* g)

{

Node\* temp;

temp = new Node;

temp->data = entry;

temp->next = g;

return temp;

}

//this function writes out the items to the output stream

ostream& operator<<(ostream& out\_s, const list& list1)

{

list::Node\* g;

if (list1.first == NULL) {

cout << " There are no values currently in the list.";

}

else {

g = list1.first;

cout << " The List is: ";

cout << " < ";

while (g != NULL) {

out\_s << g->data << " ";

g = g->next;

}

cout << " >";

return out\_s;

}

}

project6.cpp

list::~list()

{

Node\* temp;

while (first != NULL) {

temp = first;

first = first->next;

delete temp;

}

}

list::item list::a\_empty()

{

a\_a\_empty(first);

}

list::item list::a\_a\_empty(Node\* g)

{

Node\* temp;

if (g == NULL)

return 0;

else {

temp = g;

//g=g->next;

a\_a\_empty(g->next);

delete temp;

}

}

bool list::isPresent(const item& target)

{

return i\_isPresent(first, target);

}

bool list::i\_isPresent(Node\* g, const item& target)

{

if (g->data == target)

return (g != NULL);

else

return g\_isPresent(g->next, target);

}

list::Node\* list::get\_node(const item& entry, Node\* g)

{

Node\* temp;

temp = new Node;

temp->data = entry;

temp->next = g;

return temp;

}

void list::insert(const item& entry)

{

i\_insert(first, entry);

}

void list::i\_insert(Node\*& g, const item& entry)

{

if (g == NULL || entry < g->data) {

g = get\_node(entry, g);

}

else {

i\_insert(g->next, entry);

}

}

void list::remove(const item& target)

{

r\_remove(first, target);

}

void list::r\_remove(Node\*& g, const item& target)

{

Node\* temp;

if (g->data == target) {

temp = g;

g = g->next;

delete temp;

}

else {

r\_remove(g->next, target);

}

}

int list::length()

{

return l\_length(first);

}

int list::l\_length(Node\*& g)

{

if (g == NULL)

return 0;

else

return 1 + l\_length(g->next);

}

list::item list::nth(item n)

{

return n\_nth(first, n);

}

list::item list::n\_nth(Node\* g, item target)

{

int i = 1;

if (i == target)

return g->data;

else

return n\_nth(g->next, target - 1);

}

ostream& operator<<(ostream& out\_s, list& list1)

{

list::Node\* g;

g = list1.first;

while (g != NULL) {

out\_s << g->data << " ";

g = g->next;

}

return out\_s;

}

**Tests**

Test 1

csci2>g++ List.h

csci2>g++ project5.cpp

csci2>a.out

This program responds to the commands the user enters to manipulate

an ordered list of integers, which is initially empty. In the following

commands, k1 is the position in the list, and v is an integer.

e -- Re-initialize the list to be empty.

i v -- Insert the value v into the list.

r v -- Remove the value v from the list.

m -- Is the list empty?

l -- Report the length of the list.

p v -- Is the value v present in the list?

k k1 -- Report the k1th value in the list.

w -- Write out the list.

h -- See this menu.

q -- Quit.

-->i 2

-->i 4

-->i 6

-->i 8

-->i 10

-->i 12

-->e

The list is not empty.

-->w

The list is : < 2 4 6 8 10 12 >

-->r 8

-->w

The list is : < 2 4 6 10 12 >

-->p 11

The value 11 is NOT present in the list.

-->p 13

The value 13 is NOT present in the list.

-->k 9

Not the valid selection.

-->k 3

The 3th element of the list is: 10

-->k 2

The 3th element of the list is: 6

-->q

Test 2

csci2>g++ List.h

csci2>g++ project5.cpp

csci2>a.out

This program responds to the commands the user enters to manipulate

an ordered list of integers, which is initially empty. In the following

commands, k1 is the position in the list, and v is an integer.

e -- Re-initialize the list to be empty.

i v -- Insert the value v into the list.

r v -- Remove the value v from the list.

m -- Is the list empty?

l -- Report the length of the list.

p v -- Is the value v present in the list?

k k1 -- Report the k1th value in the list.

w -- Write out the list.

h -- See this menu.

q -- Quit.

-->i 6

-->i 6

-->i 6

-->i 6

-->w

The list is : < 6 >

-->l

The length of the given list is: 1

-->k 0

The 0th element of the list is: 6

-->k 6

Not the valid selection.

-->i 8

-->w

The list is : < 6,8 >

-->l

The length of the list is: 2

-->q

Test 3

csci2>g++ List.h

csci2>g++ project5.cpp

csci2>a.out

This program responds to the commands the user enters to manipulate

an ordered list of integers, which is initially empty. In the following

commands, k1 is the position in the list, and v is an integer.

e -- Re-initialize the list to be empty.

i v -- Insert the value v into the list.

r v -- Remove the value v from the list.

m -- Is the list empty?

l -- Report the length of the list.

p v -- Is the value v present in the list?

k k1 -- Report the k1th value in the list.

w -- Write out the list.

h -- See this menu.

q -- Quit.

-->w

There are no values currently in the list.

-->l

The length of the given list is zero.

-->e

The list is empty.

-->p 6

The value 6 is NOT present in the list.

-->h

This program responds to the commands the user enters to manipulate

an ordered list of integers, which is initially empty. In the following

commands, k1 is the position in the list, and v is an integer.

e -- Re-initialize the list to be empty.

i v -- Insert the value v into the list.

r v -- Remove the value v from the list.

m -- Is the list empty?

l -- Report the length of the list.

p v -- Is the value v present in the list?

k k1 -- Report the k1th value in the list.

w -- Write out the list.

h -- See this menu.

q -- Quit.

-->q

**User Document**

After completing the code, the program was compiled in putty by using the commands “g++ List.h” and “g++ project6.cpp”. Then we type a.out. The program prompts then asks for input data and we get the desired output as per our input.

**Conclusion**

We were able to design and write a class that implements an ordered list type using a linked list with recursion. I was very relieved when the program ran successfully. I had been working on this for quite some time and I hadn’t been getting the desired output.