**Project-3**

**IMPLEMENTING AN ORDERED LIST CLASS WITH A LINKED LIST**

**Name: Pawan Dhungana**

**Project no: 3**

**Due Date: September 24, 2018**

**Design Document**

**Introduction**

An ordered list is a sequence of elements, all of the same type, that is ordered by the elements' values. For example, <15, 22, 24, 29, 33, 42> is an ordered list of integers. Many kinds of data might be organized into ordered lists, and we would like to manipulate such structures in programs. This project uses a class to implement an ordered list type.

**Data Structures**

The program makes use of class named List to implement an ordered list ADT. Item is used as the data type of the items in the list using typedef. One constructor is used : List() which is a inline constructor. ~List() is a destructor that deallocates the memory used.

**Functions**

Nine functions are used in the program: make\_empty(), insert(), remove(), menu(), empty(), length(), present(), kth()and a friend function. Make\_empty() re-initializes the set to be empty, insert() insets and item into the list, remove() removes an item from the list, length() returns the size of the list; that is, the number of elements in it, present() determine if a particular integer is an element of the list, menu() displays the menu, kth() returns the kth value in the list and the friend function writes the contents of the list to output stream out\_s; that is to write out the list.

**Main Program**

The main program provides the list of choice to the user. The user can insert, remove, find length, determine if the user input element is in the list, return the kth value in the list, empty the list, write out the list and look at the menu from the main program. The main program is implemented using a switch case statement where different functions are called as per the case.

**User Document**

An ordered list is a sequence of elements, all of the same type, that is ordered by the elements' values. For example, <15, 22, 24, 29, 33, 42> is an ordered list of integers. Many kinds of data might be organized into ordered lists, and we would like to manipulate such structures in programs. This project uses a class to implement an ordered list type.

The program's name is Project3.cpp, to compile and run it, simply enter:

g++ Project3.cpp

a.out

This program responds to commands the user enters to

manipulate an ordered list of integers, which is

initially empty. In the following commands, v is any

integer, and p is a position in the list.

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.

l -- Report the length of the list.

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

k p -- Report the pth value of the list.

w -- Write out the list.

h -- See this menu.

q -- Quit.

--> i 27

--> i 42

--> i 15

--> i 33

--> i 14

--> w

List: < 14, 15, 27, 33, 42 >

--> r 33

--> w

List: < 14, 15, 27, 42 >

--> p 22

The value 22 is NOT present in the list.

--> p 42

The value 42 is present in the list.

--> k 3

The 3th element of the list is 27.

--> k 9

The list does not contain 9 values.

--> q

**Code Listing:**

#include <cstdlib>

#include <iostream>

#include <cassert>

using namespace std;

class List

{

public:

// TYPEDEF

typedef int Item;

// CONSTRUCTORS

List( ) { first = NULL; } // Inline function

// DESTRUCTOR

~List( );

// MODIFICATION MEMBER FUNCTIONS

void menu();

void make\_empty ( );

void insert ( const Item& entry );

void remove ( const Item& target );

// CONSTANT MEMBER FUNCTIONS

bool empty( ) const { return first == NULL; } // Inline function

std::size\_t length( ) const;

bool present ( const Item& target ) const;

Item kth ( std::size\_t k ) const;

// FRIEND FUNCTION for the List class:

friend std::ostream& operator << ( std::ostream& out\_s,const List& l );

private:

// DATA MEMBERS

struct Node

{

Item data;

Node \*next;

};

Node \*first;

// PRIVATE FUNCTION

Node\* get\_node ( const Item& entry, Node\* link );

};

int main()

{

List l;

l.menu();

char x;

do

{

cin>>x;

switch(x)

{

case 'e':

l.make\_empty();

cout<<"The list is now empty"<<endl;

break;

case 'i':

int number;

cin>>number;

l.insert(number);

cout<<number<<" is inserted in the list."<<endl;

break;

case 'r':

int target;

cin>>target;

l.remove(target);

cout<<target<<" is removed from the list."<<endl;

break;

case 'l':

cout<<"The list contains "<<l.length()<<" values."<<endl;

break;

case 'p':

int search;

cin>>search;

if(l.present(search)==true)

cout<<"The value "<<search<<" is present in the list."<<endl;

else

cout<<"The value "<<search<<" is not present in the list."<<endl;

break;

case 'k':

int pth;

cin>>pth;

if(pth > l.length())

cout<<"The list does not contain "<<pth<<" values."<<endl;

else

cout<<"The "<<pth<<"th value of the list is: "<<l.kth(pth)<<endl;

break;

case 'w':

cout<<"List: "<<l<<endl;

break;

case 'h':

l.menu();

break;

case 'q':

cout<<"The program exited.";

return 0;

}

}while(x!='q' || x!='Q');

}

void List::menu()

{

cout<<"This program responds to commands the user enters to";

cout<<"manipulate an ordered list of integers, which is initially empty. ";

cout<<"In the following commands, v is any integer, and p is a position in the list"<<endl;

cout<<"e -- Re-initialize the list to be empty."<<endl;

cout<<"i v -- Insert the value v into the list."<<endl;

cout<<"r v -- Remove the value v from the list."<<endl;

cout<<"l -- Report the length of the list."<<endl;

cout<<"p v-- Is the value v present in the list?"<<endl;

cout<<"k p -- Report the pth value of the list."<<endl;

cout<<"w -- Write out the list."<<endl;

cout<<"h -- See this menu."<<endl;

cout<<"q -- Quit."<<endl;

}

// Destructor

List::~List( )

{

Node\* temp;

while ( first != NULL )

{

temp = first;

first = first -> next;

delete temp;

}

}

// Modification member functions

void List::make\_empty ( )//makes the list empty

{

Node\* temp;

while ( first != NULL )

{

temp = first;

first = first -> next;

delete temp;

}

}

void List::insert ( const Item& entry )//insert value in the list

{

Node \*prev;

//assert makes sure the entered value is not already present in the list

assert ( ! present(entry) );

if ( first == NULL || entry < first->data )//to insert in the first node

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);

}

}

void List::remove ( const Item& target )//remove value from the list

{

Node \*temp;

Node \*prev;

//assert makes sure that the target value to be removed is present in the list

assert ( present(target) );

prev = first;

if ( prev->data == target )//if the target is the first element of the list

{

first = first->next;

delete prev;

}

else

{

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

prev = prev->next;

temp = prev->next;

prev->next = temp->next;

delete temp;

}

}

// Constant member functions

size\_t List::length( ) const

{

Node \*cursor;

size\_t count;

count = 0;

for ( cursor=first; cursor != NULL; cursor=cursor->next )

++count;

return count;

}

bool List::present ( const Item& target ) const//search if the value entered is present in the list

{

Node \*cursor;

for ( cursor=first;

cursor!=NULL && cursor->data!=target;

cursor=cursor->next )

;

return ( cursor != NULL );

}

List::Item List::kth ( size\_t k ) const

{

Node \*cursor;

size\_t count;

assert ( 1 <= k && k <= length() );

cursor = first;

for (count=1; count<k; ++count)

cursor = cursor->next;

return cursor->data;

}

// Friend Function

ostream& operator << ( ostream& out\_s, const List& l )

{

List::Node \*cursor;

out\_s << '<';

for ( cursor=l.first;

cursor != NULL && cursor->next != NULL;

cursor=cursor->next )

out\_s << cursor->data << ", ";

if ( cursor != NULL )

out\_s << cursor->data;

out\_s << '>';

return out\_s;

}

// Private function

List::Node\* List::get\_node ( const Item& entry, Node\* link )

{

Node \*temp;

temp = new Node;

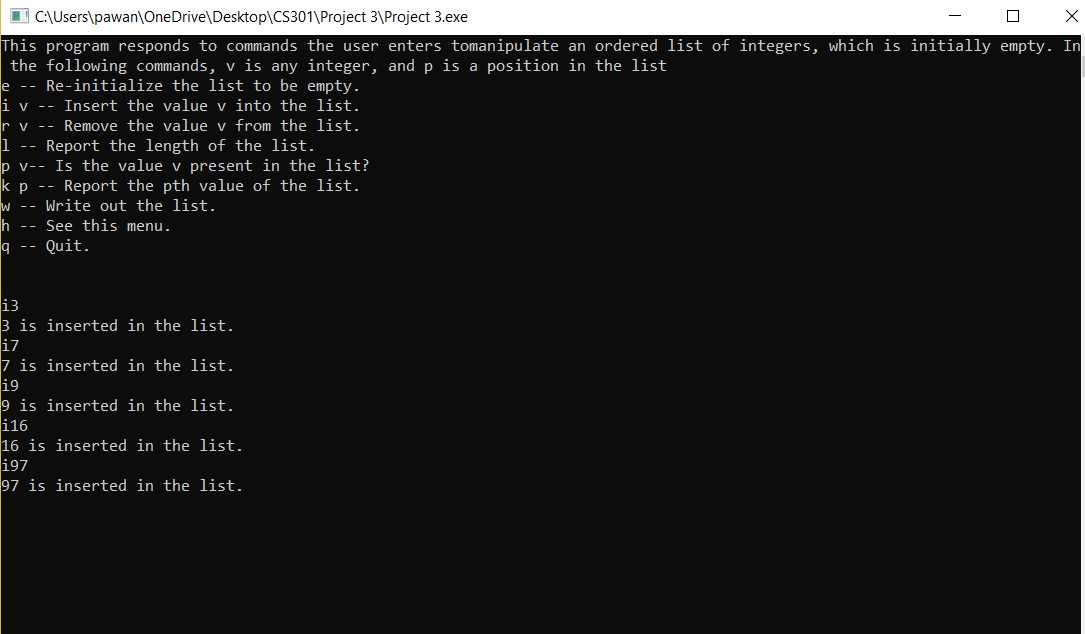
temp->data = entry;

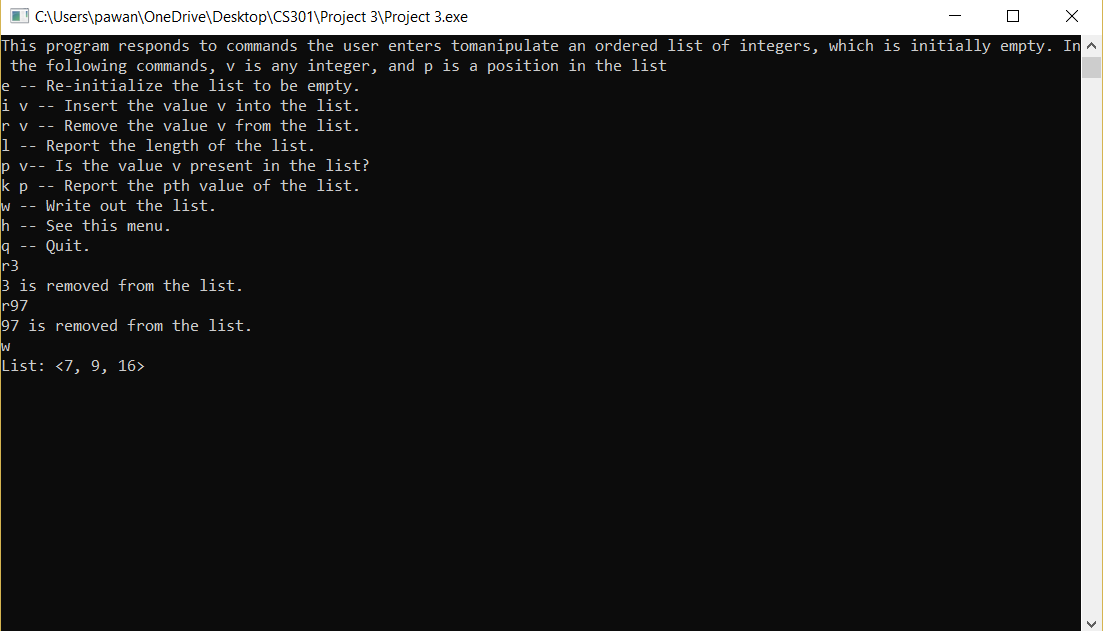
temp->next = link;

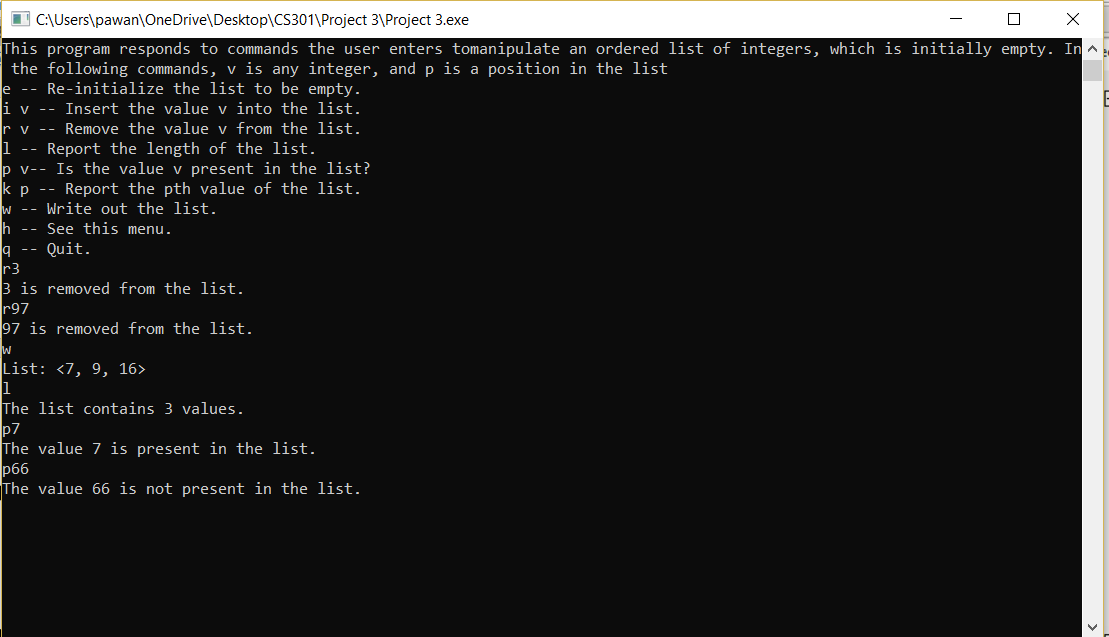
return temp;

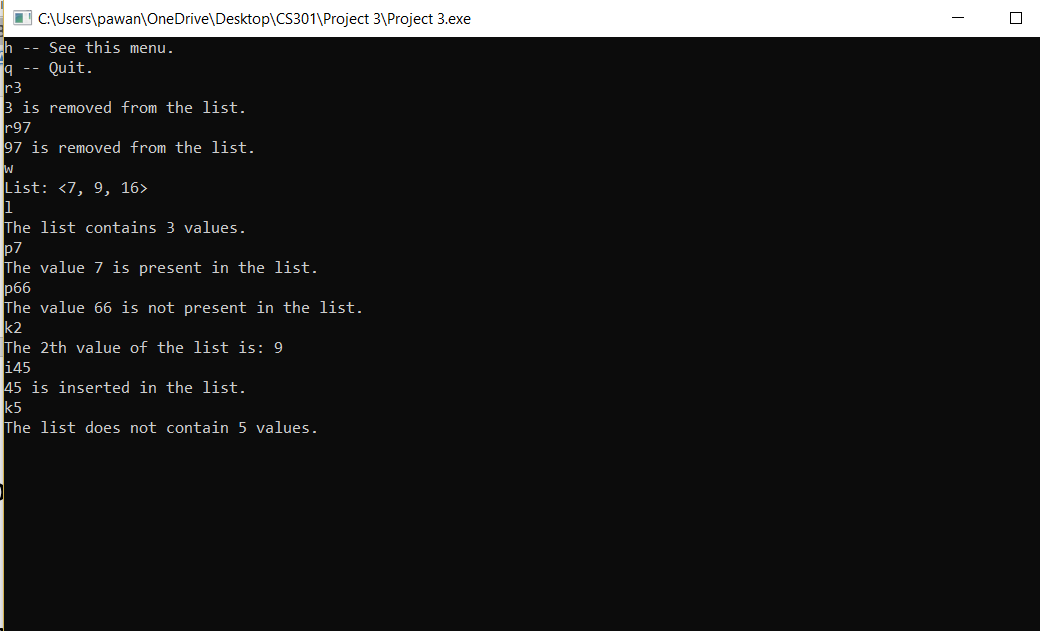
}

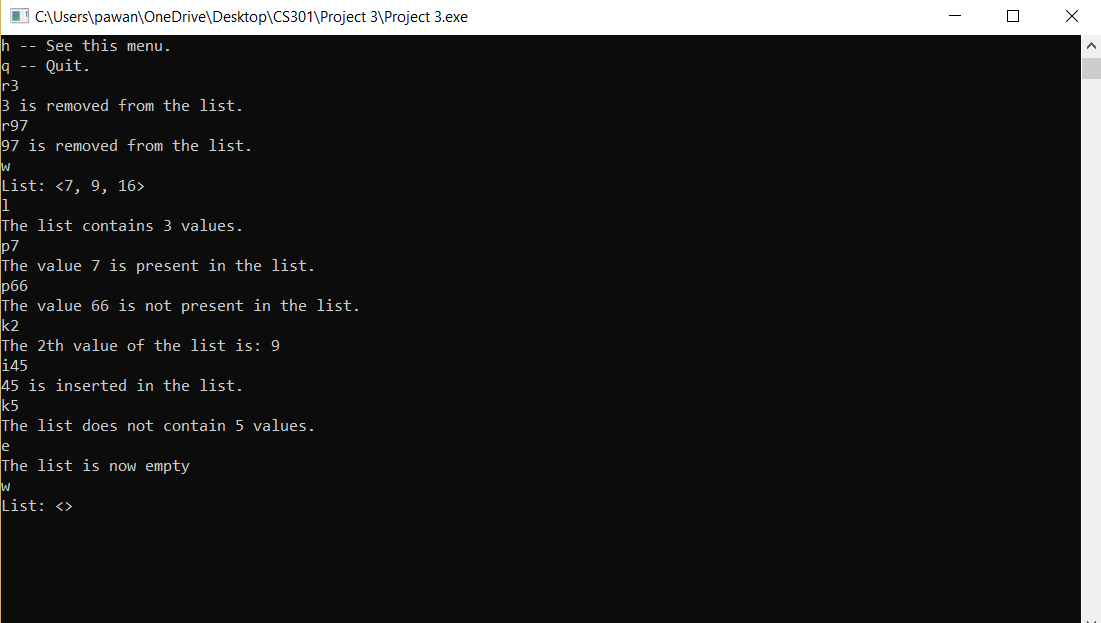
**Test Document**

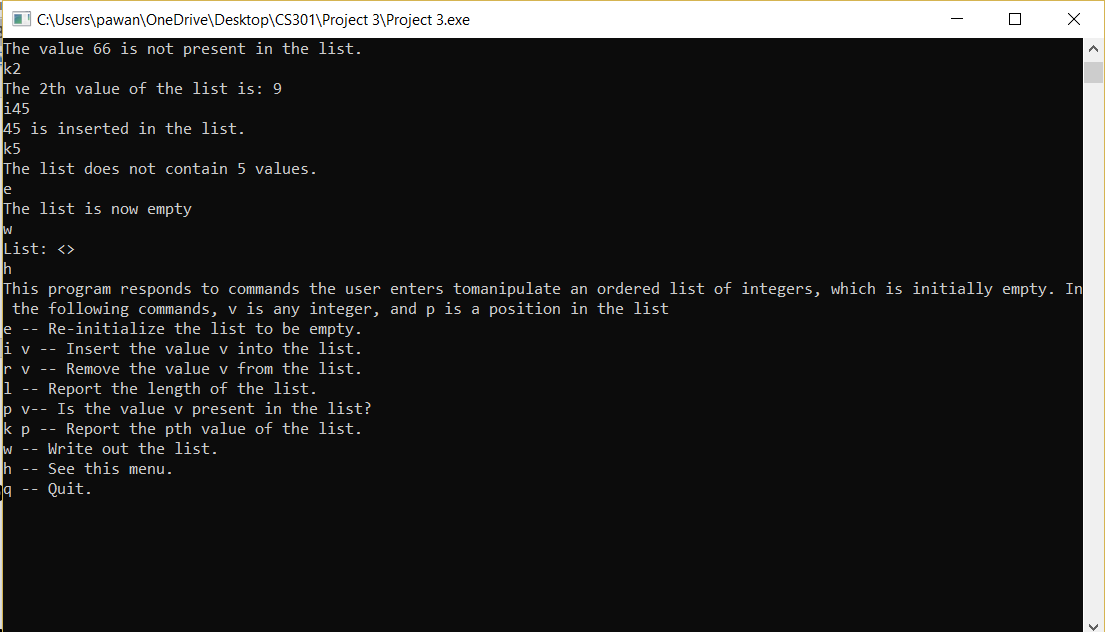
****

****

****

****

****

****

**Summary**

In this project, we implemented a program that implements an ordered list with a linked list using a class. We designed a menu-driven program to provide the user with various choices to work around with an ordered list. The user can insert, remove, find length, determine if the user input element is in the list, return the kth value in the list, empty the list, write out the list and look at the menu from the main program.

From this project, I learned how to implement an ordered list class with a linked list. This project helped me learn more clearly about how the pointers work in a C++ program.