#include <iostream>

#include <sstream>

#include <vector>

#include <string>

using namespace std;

class SubsError {

private:

int number;

public:

SubsError(int var) : number(var) {

}

int getValue() {

return number;

}

};

class IntLinkedQueue {

private:

struct Node {

int data;

Node \*next;

};

Node \*front; // -> first item

Node \*rear; // -> last item

Node \*p; // traversal position†

Node \*pp; // previous position

int size;

public:

// Default Constructor

IntLinkedQueue() : front(nullptr), rear(nullptr), size(0) {}

// Destructor; // return all dynamically allocated nodes

~IntLinkedQueue() {

Node \*next = front;

while (next) {

Node \*deleteMe = next;

next = next->next;

delete deleteMe;

}

}

//push front

void add(int val) {

Node \*n = new Node();

n->data = val;

n->next = front;

front = n;

size++;

}

//push back

void append(int n) {

p = new Node;

p->next = nullptr;

p->data = n;

(front) ? rear = rear->next = p : front = rear = p;

size++;

}

//pop front

int Remove() {

Node \*n = front;

int ret = n->data;

front = front->next;

delete n;

size--;

return ret;

}

//pop back

int remove() {

Node \*n = rear;

int ret = n->data;

rear = nullptr;

delete n;

size--;

return ret;

}

//if empty return true

bool isEmpty() {

return !(front->next);

}

//returns size of queue

int lengthOfQueue() {

return size;

}

void display() {

Node \*temp = front;

while (temp !=nullptr) {

cout << temp->data << ", ";

temp = temp->next;

}

cout << endl;

}

//clears queue

void Clear() {

Node \*next = front;

while (next) {

Node \*deleteMe = next;

next = next->next;

delete deleteMe;

size--;

}

}

};

string del = ","; // default delimiter

void menu()

{

cout << "\n--- AS9 LinkedQueue Test Menu ---\n"

<< " N - to bulk create New Queue\n"

<< " D - to Display\n"

<< " E - is Empty?\n"

<< " A - to Add ~ push front\n"

<< " a - to append ~ push back\n"

<< " R - to Remove ~ pop front\n"

<< " r - to Remove ~ pop back\n"

<< " G - to get by subscription\n"

<< " P - to put by subscription\n"

<< " L - Length of Queue?\n"

<< " C - to Clear\n"

<< " Q - to Q this program\n"

<< " H - this menu\n";

}

int main() {

bool stay = true;

// create one instance for each of the test classes

cout << "\nInstanciate an object of Queue\n";

// string catchVar; // To hold values popped off the Queue

IntLinkedQueue q;

cout << "\nFirst, test with hard-wired data sets!\n"

<< "For example, \nyou may use one set of tokens for all test classes: \n";

string choice; // user input for choices

string str; // user input for queue, delimiter, ...

int s[] = { 1, 3, 5, 7, 9 };

vector<int> input(s, end(s));

// show the initial queue

int input\_size = input.size();

for (auto item : input) {

cout << item << ((input\_size <= 1) ? " " : ", ");

input\_size--;

}

cout << endl;

// initialize the Queue class instances

for (auto i : input) q.append(i);

cout << "\nUse menu to test a Queue class instance.\n";

menu();

// main menu while

while (stay) { // main menu while starts

cout << "\n Enter your command: ";

stay = true;

cin >> choice;

cin.ignore();

int pos;

string input;

string token;

int item;

stringstream sst;

stringstream ss;

int Queue\_size;

if (choice.size() == 1) {

char ch = choice[0];

vector<string> dump;

string value;

switch (ch) { // main menu switch starts

case 'n':

case 'N':

{

// if(!q.isEmpty()) q.clear();

cout << "\t\tEnter a line of comma (,) delmited data set: ";

getline(cin, input); // user input -> string

ss << input; // string -> stream

while (getline(ss, token, ',')) { // stream -> string token

stringstream sst(token);

sst >> item;

q.append(item);

}

break;

}

case 'd':

case 'D':

{

q.display();

break;

}

case 'E':

{

if (q.isEmpty()) {

cout << "Queue is empty" << endl;

}

else {

cout << "Queue is not empty" << endl;

}

break;

}

case 'A':

{

cout << "Enter a number: ";

cin >> choice;

cout << "You have entered: " << choice << endl;

q.add(stoi(choice));

q.display();

break;

}

case 'a':

{

cout << "Enter a number: ";

cin >> choice;

cout << "You have entered: " << choice << endl;

q.append(stoi(choice));

q.display();

break;

}

case 'R':

{

int num = q.Remove();

cout << "Removed " << num << endl;

q.display();

break;

}

case 'r':

{

int num = q.remove();

cout << "Removed " << num << endl;

q.display();

break;

}

case 'L':

{

cout << "Number of Entries in Queue: " << q.lengthOfQueue();

break;

}

case 'C':

{

q.Clear();

cout << "Queue is cleared";

break;

}

case 'q': // exit Queue sub-menu

case 'Q':

stay = false;

break;

case 'h': // help - display menu

case 'H':

case '?':

menu();

break;

default:

cout << "\nInvalid Selection!\nTry again!\n\n";

} // end of menu switch

} // only process single character

} // end of main menu while

cout << "Program exit!\n";

return 0;

system("pause");

} // end of main