**All the files are in this document as well:**

**Stack.h:**

#pragma once

#ifndef stack\_h

#define stack\_h

#include<iostream>

#include<fstream>

using namespace std;

class Node {

public:

string data;

Node\* next;

Node\* prev;

Node(string d) { data = d; next = NULL; }

};

class Stack {

public:

//bool append(string d);

void push(string name); // pushes cars to lanes

int search(string name); // searches if the car is in any of the lanes

int searchv2(string name); // different from the first search returns an index of how far the car that wants to leave is

string pop(string name); // pops car from lane

void display(); // displays a lane of cars

Node\* head = NULL;

Node\* tail = NULL;

};

#endif

**Garage.h**

#pragma once

#ifndef garage\_h

#define garage\_h

#include<iostream>

#include<fstream>

#include "Stack.h"

using namespace std;

class Garage {

public:

Stack lane1;

Stack lane2;

Stack Street;

void readCars();

};

class Car {

public:

int numOfMoves = 0;

string op;

string name;

};

#endif

**Source.cpp**

#include "Garage.h"

#include "Stack.h"

int main()

{

Garage Stack;

Stack.readCars(); // does basically everything

return 0;

}

/\* example list to use

A 123DEF

A 345XYZ

D 123DEF

A 674GTX

A 896YUX

D 234FDS

A 567TYD

A 891JKL

D 345XYZ

A 786IOC

A 102931

A 123ABC

A 345XYZ

D 896YUX

D 674GTX

A 896YUX

D 123ABC

A 567TYD

D 567TYD

A 786IOC

A 102931

A 123IOS

A 66DADD

D ONGOD1

A KILLER

A BIGROL

D KILLER

A LOWROL

A LOLOLO

D LOLOLO

A JOCKY1

A GDADzZ

A STACKZ

A BIZZNE

D CORNOP

D BIZNEE

A MAZAEO

D MAZAEO

A 111111

D STACKZ

A 786IOC

A 102931

\*/

/\*

A ab1

A ab2

A ab3

A ab4

A ab5

A ab6

A ab7

A ab8

A ab9

A ab10

A ab11

A ab12

A ab13

A ab14

A ab15

D ab2

A ab16

A ab17

A ab18

A ab19

A ab20

A ab21

A ab22

A ab23

A ab24

\*/

**Garage.cpp**

#include "Garage.h"

#include "Stack.h"

void Garage::readCars()

{

ifstream read;

Stack lane1;

Stack lane2;

Stack street;

Car carData; // stores operation and license plate

int lane1Count = 0, lane2Count = 0;

int tempCar; // used to see if a car is already in garage

int index = 0;

string temp; // used for putting cars back into the lanes after being put into the street

read.open("ParkingGarage.txt");

if (!read.is\_open()) {

cout << "Error opening file make sure it is the exact name and it is in the correct folder " << endl;

return;

}

while (!read.eof()) {

read >> carData.op >> carData.name;

cout << "New Car: " << endl;

cout << "Read in " << carData.op << " and " << carData.name << endl;

if (carData.op == "A") { // if a car wants to enter the garage

if (lane1Count != 10) { // if lane 1 is not full

tempCar = lane1.search(carData.name); // search if the car is already in lane 1

if (tempCar == 0) { // if car is already in lane 1

cout << "Car is already in garage: lane 1" << endl;

}

else { // if car is not in the garage

lane1.push(carData.name);

lane1Count++;

cout << "Pushed " << carData.name << endl << endl;

}

}

else if (lane2Count != 10) { // if lane 2 is not full

tempCar = lane2.search(carData.name); // search if the car is already in lane 2

if (tempCar == 0) { // if car is already in lane 2

cout << "Car is already in garage: lane 2" << endl;

}

else {

lane2.push(carData.name);

lane2Count++;

cout << "Pushed " << carData.name << endl << endl;

}

}

else { // if both lanes are full

cout << "Both lanes full " << carData.name << " is leaving" << endl << endl;

}

}

else { // carData.op == 'D'

tempCar = lane1.search(carData.name); // searching if car is not in lane 1

if (tempCar == -1) {

//cout << "Car is not in the lane 1 checking lane 2 " << endl;

tempCar = lane2.search(carData.name); // searching if car is in lane 2

if (tempCar == -1) { // if car is not in garage

cout << "Car is not in lane 1 or 2, the car is not in the garage" << endl << endl;

}

else { // if car is found in lane 2

index = lane2.searchv2(carData.name); // return how many cars are blocking it

for (int i = 0; i <= index; i++) { // pop each car blocking it and the car we are trying to remove

temp = lane2.pop(carData.name);

if (temp != carData.name && temp != "0") { // push cars to the street that isn't the car we are trying to remove

Street.push(temp);

}

//cout << "temp = " << temp << endl;

}

carData.numOfMoves++;

for (int i = 0; i < index; i++) { // pushing cars back into lane

temp = Street.pop(carData.name); // pop cars from street return the car that got popped

lane2.push(temp); // push into lane 2

}

lane2Count--;

cout << carData.name << " removed from lane 2 and was moved " << carData.numOfMoves << " Times" << endl << endl;

}

}

else { // if car is found in lane 1 same process

index = lane1.searchv2(carData.name);

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

temp = lane1.pop(carData.name);

if (temp != carData.name && temp != "0") {

Street.push(temp);

}

//cout << "temp = " << temp << endl;

}

carData.numOfMoves++;

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

temp = Street.pop(carData.name);

lane1.push(temp);

}

lane1Count--;

cout << carData.name << " removed from lane 1 and was moved " << carData.numOfMoves << " Times" << endl << endl;

}

}

}

cout << "\nNo more cars here are the final results " << endl;

cout << "Lane 1: "; lane1.display();

cout << "Lane 2: "; lane2.display();

cout << "Street: "; Street.display();

return;

}

**Stack.cpp**

#include "Stack.h"

#include "Garage.h"

void Stack::push(string name)

{

Node\* newNode = new Node(name);

if (!newNode)

return;

// list is empty

if (head == NULL) {

head = newNode;

tail = newNode;

}

else {// list is not empty

newNode->next = head;

head = newNode;

}

return;

}

void Stack::display() {

if (head == NULL) {//list is empty

cout << "List is empty" << endl;

return;

}

//list is not empty

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << ' ';

temp = temp->next;

}

cout << endl;

}

int Stack::search(string carName)

{

Node\* temp = head;

int index = 0;

if (head == NULL) {

return -1;

}

while (temp != NULL) {

if (temp->data == carName) {

return 0;

}

temp = temp->next;

index++;

}

return -1; // not found

}

string Stack::pop(string carName)

{

int counter = 0;

Stack Street;

//list is empty

if (head == NULL) {

return "0";

}

// only 1 node in the list

if (head->next == NULL && head->data == carName) {

delete head;

head = NULL;

tail = NULL;

return "0";

}

//more then 1 node

Node\* temp;

string save;

save = head->data;

temp = head;

head = head->next;

delete temp;

return save;

}

int Stack::searchv2(string carName) {

Node\* temp = head;

int index = 0;

if (head == NULL) {

return -1;

}

while (temp != NULL) {

if (temp->data == carName) {

return index;

}

temp = temp->next;

index++;

}

return -1; // not found

}