//SOURCE CPP

```
#include "Garage.h"
#include "Stack.h"
#include <iostream>
#include <string>
#include <fstream>

using namespace std;
// main function
int main()
{
	Garage<car> Garage;
	//read in the file
	fstream input;
	//open the text file
	input.open("Garage.txt");
	if (!input)
	{
		cout << "ERROR ! file can not open !" << endl;
		input.close();
		system("pause");
		return 0;
	}
	cout << " \n     WELCOME  TO  THE  PARKING  GARAGE !!!!  " << endl;
	cout << "............................................. " << endl << endl;

	while (!input.eof())
	{
		car newcar;//make a newcar
		input >> newcar.code >> newcar.Name;//to get the car's status and the plate
number
		//find the car status is departing or arriving.
		switch (newcar.code)
		{
		case 'A':
			Garage.arrive(newcar);
			break;
		case 'D':
			Garage.depart(newcar);
			break;
		default:
			cout << "This car doesn't exist" << endl;
			break;
		}
	}
	system("pause");
	return 0;
}
```

//STACK CPP

```
#include"stack.h"
using namespace std;

struct car
{
	int counter = 0;
	char code;
	string Name;
	int number;
	bool operator==(const car& obj2)
	{
		if (Name == obj2.Name)
			return true;
		else
			return false;
	}
};
```

//GARAGE CPP

```
#include<iomanip>

#include"stack.h"
using namespace std;

struct car
{
	int counter = 0;
	char code;
	string Name;
	int number;
	bool operator==(const car& obj2)
	{
		if (Name == obj2.Name)
			return true;
		else
			return false;
	}
};
```

//GARAGE HEADER

```
#ifndef Garageh
#define Garageh

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

#include"stack.h"
using namespace std;

struct car
{
	int counter = 0;
	char code;
	string Name;
	int number;
	bool operator==(const car& obj2)
	{
		if (Name == obj2.Name)
			return true;
		else
			return false;
	}
};

template<class GarageType>
class Garage
{
public:
	Garage();
```

```
        bool isEmpty() const;
        bool isFull() const;
        void arrive(car v);
        void depart(car v);
        int searchCar(car v);
private:
        Stack<car> line1;
        Stack<car> line2;
        Stack<car> street;
        car array[100];
        int carcount = 0;
};

template<class GarageType>
Garage<GarageType>::Garage()
{
        Stack<car> line1();
        Stack<car> line2();
        Stack<car> street(100);
}

template<class GarageType>
int Garage<GarageType>::searchCar(car v)
{
        for (int i = 0; i < 100; i++)
        {
                if (array[i].Name == v.Name)
                {
                        return (array[i].number);
                }
        }
        return -1;
}

template<class GarageType>
void Garage<GarageType>::depart(car v)
{
        int carposition = searchCar(v);
        car temp;
        if (carposition == 1)
        {
                while (!line1.IsEmpty())
                {
                        if (line1.Top().Name == v.Name)
```

```
                    {
                            v.counter = line1.Top().counter;
                            line1.pop();
                            cout << endl;
                            cout << v.Name << " is taken out from the parking garage" <<
endl;
                            cout << v.Name << " was moved " << v.counter << " times" <<
endl;
                            cout << endl;
                            break;
                    }
                    else
                    {
                            temp = line1.Top();
                            line1.pop();
                            temp.counter++;
                            street.push(temp);
                    }
            }
            while (!street.IsEmpty())
            {
                    temp = street.Top();
                    street.pop();
                    temp.counter++;
                    line1.push(temp);
            }
    }
    else if (carposition == 2)
    {
            while (!line2.IsEmpty())
            {
                    if (line2.Top().Name == v.Name)
                    {
                            v.counter = line1.Top().counter;
                            line2.pop();
                            cout << endl;
                            cout << v.Name << " IS BEING  TAKEN OUT  FROM  THE
PARKING  GARAGE !!!!!" << endl;
                            cout << v.Name << " WAS MOVED IN  " << v.counter << " TIMES
" << endl;
                            cout << endl;
                    }
                    else
                    {
                        temp = line2.Top();
                        line2.pop();
                        temp.counter++;
                        street.push(temp);
                    }
                }
                while (!street.IsEmpty())
                {
                    temp = street.Top();
                    street.pop();
                    temp.counter++;
                    line2.push(temp);
                }
        }
        else
        {
                cout << "Car isn't parked here" << endl;
                cout << endl;
        }
}
template<class GarageType>
void Garage<GarageType>::arrive(car v)
{
        if (!line1.IsFull())
        {
                line1.push(v);
                v.number = 1;
                cout << left << setw(10) << v.Name << "HAS BEEN PARKED IN THE LANE  ---->
1" << endl;
                cout << endl;
                array[carcount] = v;
                carcount++;
        }
        else if (!line2.IsFull())
        {
                line2.push(v);
                v.number = 2;
                cout << left << setw(10) << v.Name << "HAS BEEN PARKED IN THE LANE  ---->
2" << endl;
                cout << endl;
                array[carcount] = v;
                carcount++;
        }
        else
```

```
	{
		cout << "SORRY!!... BOTH LANE 1 AND LANE 2 ARE FULL.  NO MORE
SPACES LEFT. " << endl;
	}
}
#endif
#pragma once
```

//STACK HEADER

```
#ifndef Stackh
#define Stackh

#include<cstdlib>
#include<iostream>
using namespace std;

template<class StackType>
class Stack
{
public:
	Stack(); // DEFAULT  CONSTRUCTOR
	Stack(int MaxStackSize); // CONSTRUCTOR
	~Stack();
	bool IsEmpty() const;
	bool IsFull() const;
	StackType Top() const;
	void push(const StackType& x);
	void pop();
private:
	int top; //  CURRENT  TOP OF STACK
	int MaxTop;
	StackType* stack; // ELEMENT
};

template<class StackType>
Stack<StackType>::Stack()
{
	MaxTop = 10;
	stack = new StackType[MaxTop];
	top = -1;
}
```

```
template<class StackType>
Stack<StackType>::Stack(int MaxStackSize)
{
	MaxTop = MaxStackSize - 1;
	stack = new StackType[MaxStackSize];
	top = -1;
}

template<class StackType>
Stack<StackType>::~Stack()
{
	delete[] stack;
}

template<class StackType>
bool Stack<StackType>::IsEmpty() const
{
	if (top == -1)
		return true;
	else
		return false;
}

template<class StackType>
bool Stack<StackType>::IsFull() const
{
	if (top == MaxTop - 1)
		return true;
	else
		return false;
}

template<class StackType>
StackType Stack<StackType>::Top() const
{
	if (IsEmpty())
	{
		cout << "The stack is empty" << endl;
	}
	else
	{
		return stack[top];
	}
}
```

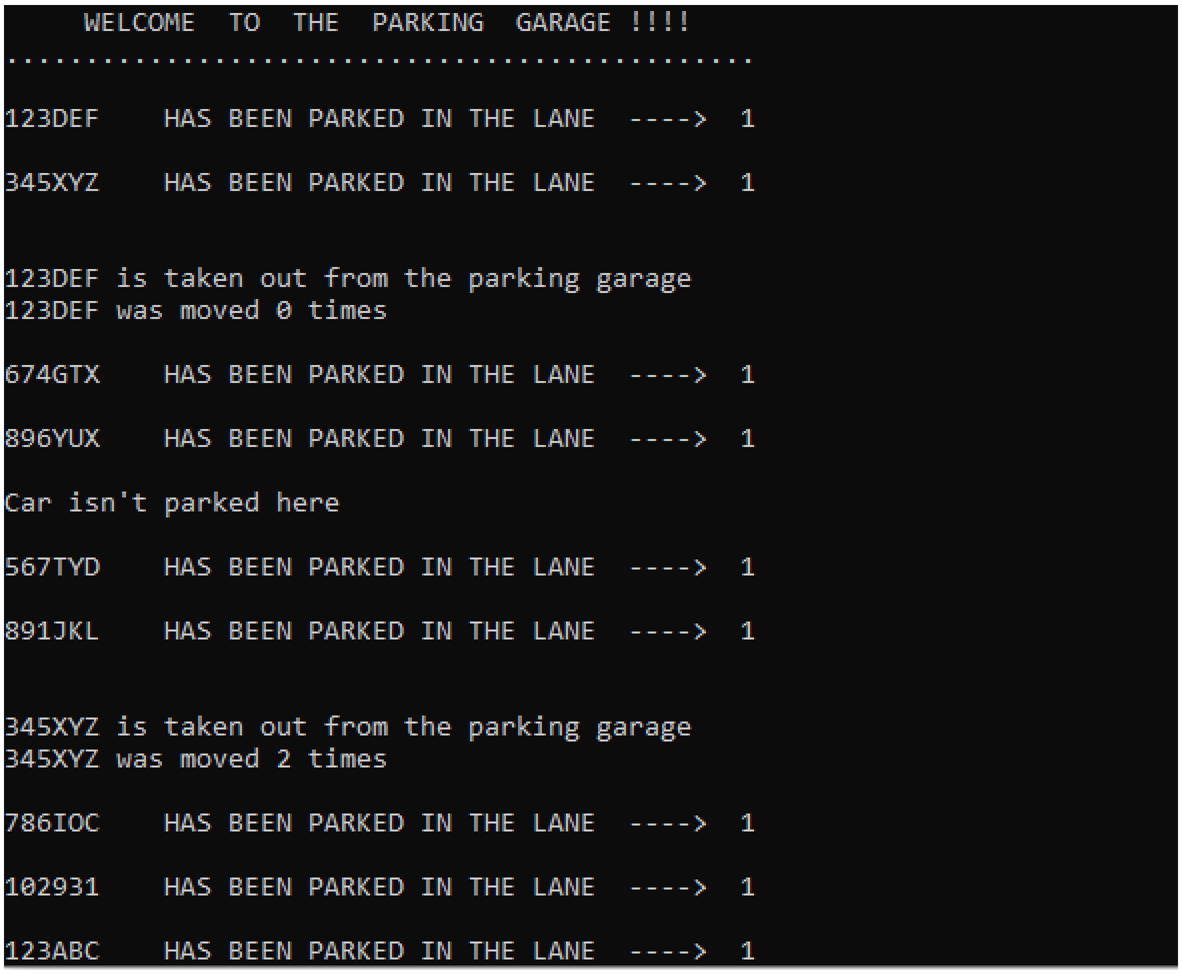
```
template<class StackType>
void Stack<StackType>::push(const StackType& x)
{
	if (IsFull())
	{
		cout << "The stack is full" << endl;
	}
	else
	{
		top++;
		stack[top] = x;
	}
}

template<class StackType>
void Stack<StackType>::pop()
{
	if (IsEmpty())
	{
		cout << "The stack is empty" << endl;
	}
	else
	{
		top--;
	}
}
#endif
```

//GARAGE TXT

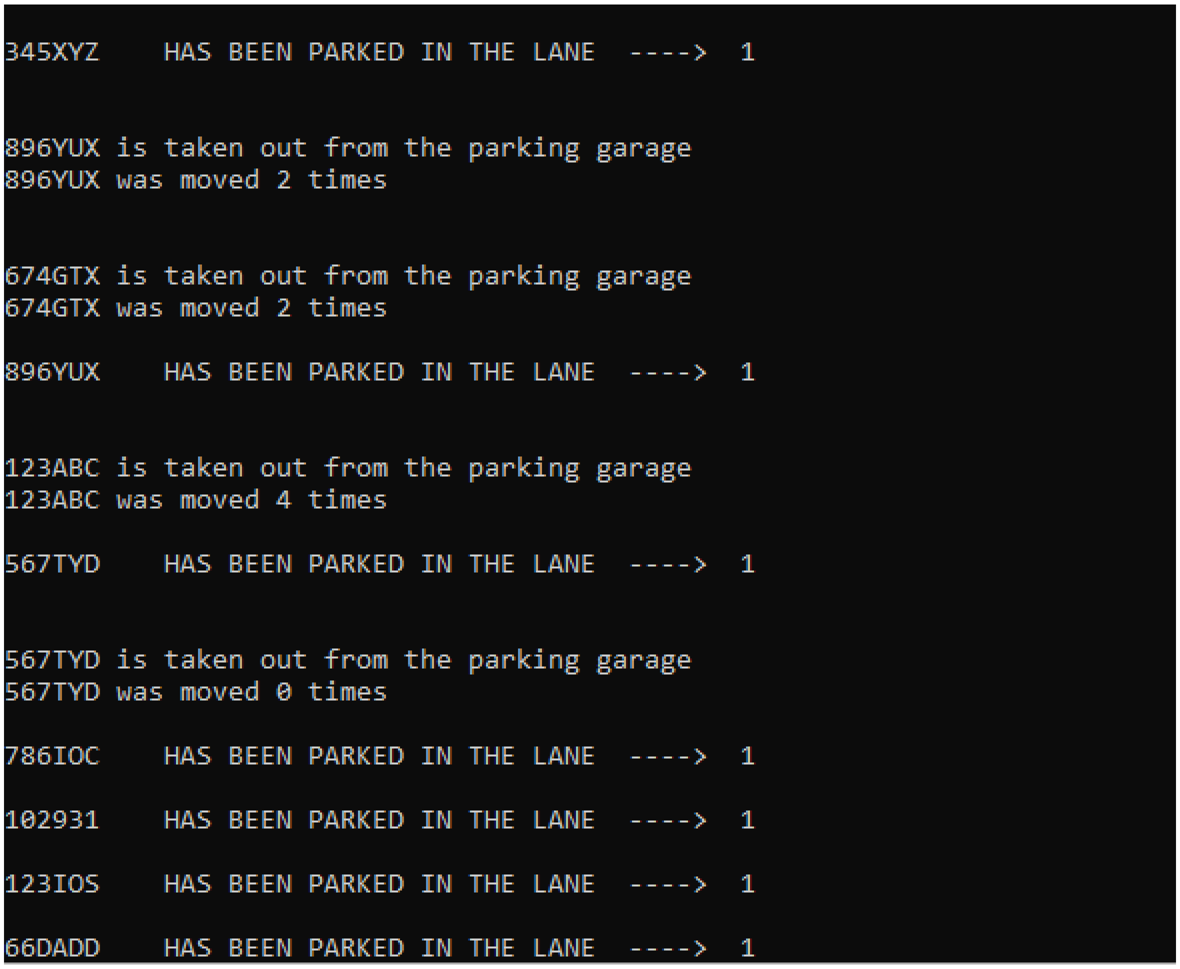
```
A 123DEF
A 345XYZ
D 123DEF
A 674GTX
A 896YUX
D 234FDS
A 567TYD
A 891JKL
D 345XYZ
A 786IOC
A 102931
```

```
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
```

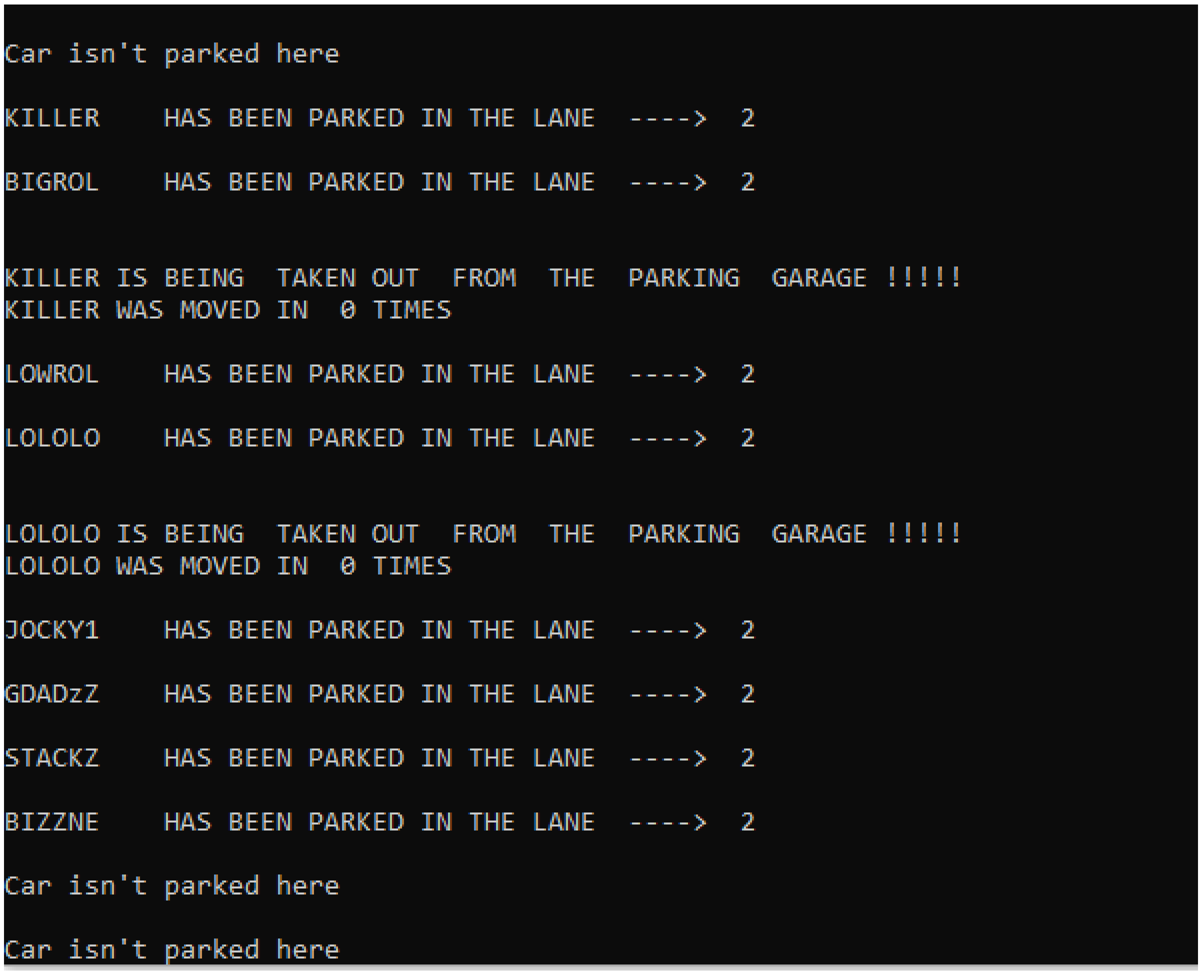
Output



```
     WELCOME   TO   THE   PARKING   GARAGE !!!!
...............................................

123DEF    HAS BEEN PARKED IN THE LANE  ---->  1

345XYZ    HAS BEEN PARKED IN THE LANE  ---->  1


123DEF is taken out from the parking garage
123DEF was moved 0 times

674GTX    HAS BEEN PARKED IN THE LANE  ---->  1

896YUX    HAS BEEN PARKED IN THE LANE  ---->  1

Car isn't parked here

567TYD    HAS BEEN PARKED IN THE LANE  ---->  1

891JKL    HAS BEEN PARKED IN THE LANE  ---->  1


345XYZ is taken out from the parking garage
345XYZ was moved 2 times

786IOC    HAS BEEN PARKED IN THE LANE  ---->  1

102931    HAS BEEN PARKED IN THE LANE  ---->  1

123ABC    HAS BEEN PARKED IN THE LANE  ---->  1
```



```
345XYZ    HAS BEEN PARKED IN THE LANE  ---->  1


896YUX is taken out from the parking garage
896YUX was moved 2 times


674GTX is taken out from the parking garage
674GTX was moved 2 times

896YUX    HAS BEEN PARKED IN THE LANE  ---->  1


123ABC is taken out from the parking garage
123ABC was moved 4 times

567TYD    HAS BEEN PARKED IN THE LANE  ---->  1


567TYD is taken out from the parking garage
567TYD was moved 0 times

786IOC    HAS BEEN PARKED IN THE LANE  ---->  1

102931    HAS BEEN PARKED IN THE LANE  ---->  1

123IOS    HAS BEEN PARKED IN THE LANE  ---->  1

66DADD    HAS BEEN PARKED IN THE LANE  ---->  1
```



```
Car isn't parked here

KILLER    HAS BEEN PARKED IN THE LANE  ---->  2

BIGROL    HAS BEEN PARKED IN THE LANE  ---->  2


KILLER IS BEING  TAKEN OUT  FROM  THE  PARKING  GARAGE !!!!!
KILLER WAS MOVED IN  0 TIMES

LOWROL    HAS BEEN PARKED IN THE LANE  ---->  2

LOLOLO    HAS BEEN PARKED IN THE LANE  ---->  2


LOLOLO IS BEING  TAKEN OUT  FROM  THE  PARKING  GARAGE !!!!!
LOLOLO WAS MOVED IN  0 TIMES

JOCKY1    HAS BEEN PARKED IN THE LANE  ---->  2

GDADzZ    HAS BEEN PARKED IN THE LANE  ---->  2

STACKZ    HAS BEEN PARKED IN THE LANE  ---->  2

BIZZNE    HAS BEEN PARKED IN THE LANE  ---->  2

Car isn't parked here

Car isn't parked here
```