```cpp
// Assignment 8
// Brady Feng (Partner A)
// Ivan Chen (Partner B)

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

const int NEW_DATA_RANGE = 30;
const int MAX_PARKING = 50;

//PART B
void  populateParking(ifstream & fin, string names[MAX_PARKING], int isStaff[MAX_PARKING]
)
{

    string currName="";
    int personType, spotNum = -1;

    while(fin>>personType>>currName>>spotNum)
    {
        int i=spotNum-101;

        isStaff[i]=personType;
        names[i]=currName;

    }
}

// PART C
void readNewData(ifstream & fin, int isStaff[NEW_DATA_RANGE], string
names[NEW_DATA_RANGE])
{
    int staff = 0;
    string name = "";
    int i = 0;
    while(fin >> staff >> name)
    {
        isStaff[i] = staff;
        names[i] = name;
        i++;
    }
}

//PART D
void clearSpot(string names[MAX_PARKING],int isStaff[MAX_PARKING], string deleteName, int
status)
{
    for(int i = 0; i<MAX_PARKING; i++)
    {
        if(names[i]==deleteName)
        {
            names[i]=" ";
```

```
            isStaff[i]=-1;
        }
    }
}

// PART E
int nextValidParking(int isStaff[MAX_PARKING], int status)
{
    for(int i = 0; i < MAX_PARKING; i++)
    {
        if(isStaff[i] == -1)
        {

            if(i < 25 && status == 1)
                return i;

            else if(i >= 25)
                return i;

        }
    }
    return -1;
}

//PART F
bool addName(string names[MAX_PARKING], int isStaff[MAX_PARKING],string addName, int
status)
{
    int nextSpot = nextValidParking(isStaff, status);

    if(nextSpot == -1)
    {
         return false;
    }

    else
    {
        isStaff[nextSpot]=  status;
        names[nextSpot]=addName;
    }


    return true;
}

// PART G
void rearrange(int isStaff[NEW_DATA_RANGE], string names[NEW_DATA_RANGE])
{
    for(int i = 25; i < 50; i++)
    {
        if(isStaff[i] == 1)
        {
            string name = names[i];
            int staff = isStaff[i];
            clearSpot(names, isStaff, names[i], isStaff[i]);
```

```
            addName(names, isStaff, name, staff);
        }
    }
}


// PART H
void output(ofstream & fout, int isStaff[NEW_DATA_RANGE], string names[NEW_DATA_RANGE])
{
    for(int i = 0; i < MAX_PARKING; i++)
    {
        fout << i + 101;

        if(isStaff[i] == -1)
            fout << setw(25) << "Empty" << endl;
        else
            fout << setw(25) << names[i] << setw(5) << isStaff[i] << endl;
    }
    fout << endl;
}

int main()
{
    ifstream parking_current("parking_current.txt");
    ifstream parking_remove("parking_remove.txt");
    ifstream parking_add("parking_add.txt");

    if(!parking_current || !parking_remove || !parking_add)
    {
        cout << "File not found :(" << endl;
        return EXIT_FAILURE;
    }

    int facultyOrStudent[MAX_PARKING] = {};
    for(int i = 0; i < MAX_PARKING; i++)
        facultyOrStudent[i] = -1;

    string names[MAX_PARKING] = {};

    //PART I
    ofstream outputA("outputA.txt");

    //state a)
    populateParking(parking_current, names, facultyOrStudent);
    outputA<<"Initial Parking Lot:"<<endl;
    output(outputA, facultyOrStudent, names);



    //state b)

    int addingIsStaff[NEW_DATA_RANGE]={};
    for(int i = 0; i < NEW_DATA_RANGE; i++)
        addingIsStaff[i] = -1;
    string addingNames[NEW_DATA_RANGE]={};
```

```
    int removingIsStaff[NEW_DATA_RANGE]={};
    string removingNames[NEW_DATA_RANGE]={};

    readNewData(parking_remove, removingIsStaff, removingNames);
    readNewData(parking_add, addingIsStaff, addingNames);

    for(int i = 0; i<NEW_DATA_RANGE; i++)
    {
        clearSpot(names, facultyOrStudent, removingNames[i], removingIsStaff[i]);
    }
    rearrange(facultyOrStudent,names);


    outputA<<"Removed and Reassigned Parking Lot:"<<endl;
    output(outputA, facultyOrStudent, names);

    //state c)
    outputA<<"Final Parking Lot:"<<endl;
    for(int i = 0; i<NEW_DATA_RANGE; i++)
    {
        if(addName(names, facultyOrStudent, addingNames[i], addingIsStaff[i])==false &&
addingIsStaff[i] != -1)
        {
            outputA<<"Unable to find spot for "<<" "<<addingNames[i]<<endl;
        }

    }
    output(outputA, facultyOrStudent, names);

    outputA.close();
    parking_current.close();
    parking_remove.close();
    parking_add.close();
}
```