```cpp
#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;
        }
    }
}
```

```
53:
54: // PART E
55: int nextValidParking(int isStaff[MAX_PARKING], int status)
56: {
57:     for(int i = 0; i < MAX_PARKING; i++)
58:     {
59:         if(isStaff[i] == -1)
60:         {
61:
62:             if(i < 25 && status == 1)
63:                 return i;
64:
65:             else if(i >= 25)
66:                 return i;
67:
68:         }
69:     }
70:     return -1;
71: }
72:
73: //PART F
74: bool addName(string names[MAX_PARKING], int isStaff[MAX_PARKING],string addName, int
    status)
75: {
76:     int nextSpot = nextValidParking(isStaff, status);
77:
78:     if(nextSpot == -1)
79:     {
80:          return false;
81:     }
82:
83:     else
84:     {
85:         isStaff[nextSpot]=  status;
86:         names[nextSpot]=addName;
87:     }
88:
89:
90:     return true;
91: }
92:
93: // PART G
94: void rearrange(int isStaff[NEW_DATA_RANGE], string names[NEW_DATA_RANGE])
95: {
96:     for(int i = 25; i < 50; i++)
97:     {
98:         if(isStaff[i] == 1)
99:         {
100:             string name = names[i];
101:             int staff = isStaff[i];
102:             clearSpot(names, isStaff, names[i], isStaff[i]);
103:             addName(names, isStaff, name, staff);
104:         }
105:     }
106: }
```

```
107:
108:
109: // PART H
110: void output(ofstream & fout, int isStaff[NEW_DATA_RANGE], string
     names[NEW_DATA_RANGE])
111: {
112:     for(int i = 0; i < MAX_PARKING; i++)
113:     {
114:         cout << i + 101;
115:
116:         if(isStaff[i] == -1)
117:             cout << setw(25) << "Empty" << endl;
118:         else
119:             cout << setw(25) << names[i] << setw(5) << isStaff[i] << endl;
120:     }
121: }
122:
123: int main()
124: {
125:     ifstream parking_current("parking_current.txt");
126:     ifstream parking_remove("parking_remove.txt");
127:     ifstream parking_add("parking_add.txt");
128:
129:     if(!parking_current || !parking_remove || !parking_add)
130:     {
131:         cout << "File not found :(" << endl;
132:         return EXIT_FAILURE;
133:     }
134:
135:     int facultyOrStudent[MAX_PARKING] = {};
136:     for(int i = 0; i < MAX_PARKING; i++)
137:         facultyOrStudent[i] = -1;
138:
139:     string names[MAX_PARKING] = {};
140:
141:     //PART I
142:     ofstream outputA("outputA.txt");
143:
144:     //state a)
145:     populateParking(parking_current, names, facultyOrStudent);
146:     cout<<"Initial Parking Lot:"<<endl;
147:     output(outputA, facultyOrStudent, names);
148:
149:
150:
151:     //state b)
152:
153:     int addingIsStaff[NEW_DATA_RANGE]={};
154:     for(int i = 0; i < NEW_DATA_RANGE; i++)
155:         addingIsStaff[i] = -1;
156:     string addingNames[NEW_DATA_RANGE]={};
157:
158:     int removingIsStaff[NEW_DATA_RANGE]={};
159:     string removingNames[NEW_DATA_RANGE]={};
160:
```

```
161:     readNewData(parking_remove, removingIsStaff, removingNames);
162:     readNewData(parking_add, addingIsStaff, addingNames);
163:
164:     for(int i = 0; i<NEW_DATA_RANGE; i++)
165:     {
166:         clearSpot(names, facultyOrStudent, removingNames[i], removingIsStaff[i]);
167:     }
168:     rearrange(facultyOrStudent,names);
169:
170:
171:     cout<<"Removed and Reassigned Parking Lot:"<<endl;
172:     output(outputA, facultyOrStudent, names);
173:
174:     //state c)
175:     cout<<"Final Parking Lot:"<<endl;
176:         for(int i = 0; i<NEW_DATA_RANGE; i++)
177:     {
178:         if(addName(names, facultyOrStudent, addingNames[i], addingIsStaff[i])==false
    && addingIsStaff[i] != -1)
179:         {
180:             cout<<"Unable to find spot for "<<" "<<addingNames[i]<<endl;
181:         }
182:
183:     }
184:     output(outputA, facultyOrStudent, names);
185:
186: }
```