```
 1: //Kalli Bonin and Ethan Goold
 2: //Question 1 - Parking Services
 3:
 4: #include <iostream>
 5: #include <cmath>
 6: #include <cstdlib>
 7: #include <fstream>
 8: #include <iomanip>
 9:
10: using namespace std;
11:
12: const int NUM_SPOTS = 51;
13:
14: void read_current(ifstream & fin, string names[], int occupancy[])
15: {
16:     int count = 0;
17:     int tempOccp = -1;
18:     string tempName = "";
19:     int spotNum = 0;
20:
21:     while (count < NUM_SPOTS && fin >> tempOccp)
22:     {
23:         fin >> tempName >> spotNum;
24:
25:         names[spotNum] = tempName;
26:         occupancy[spotNum] = tempOccp;
27:     }
28: }
29:
30: int remove_add(ifstream & fin, string RAnames[], int RAoccupancy[])
31: {
32:     int index = 0;
33:     int status = 0;
34:     string name = "";
35:
36:     while (fin >> status)
37:     {
38:         fin >> name;
39:         RAoccupancy[index] = status;
40:         RAnames[index] = name;
41:         index++;
42:     }
43:     return index;
44: }
45:
46: void free_up_space(string names[], int occupancy[], string name)
47: {
48:     for (int i = 0; i < NUM_SPOTS; i++)
49:     {
50:         if (name == names[i])
```

```
 51:         {
 52:             names[i] = "";
 53:             occupancy[i] = -1;
 54:         }
 55:     }
 56: }
 57:
 58: int free_space(int occupancy[])
 59: {
 60:     for (int i = 1; i < NUM_SPOTS; i++)
 61:     {
 62:         if (occupancy[i] == -1)
 63:             return i;
 64:         //all parking spots are full
 65:         else
 66:             return -1;
 67:     }
 68: }
 69:
 70: bool assign_space (string names[], int occupancy[], string name, int status)
 71: {
 72:     int bestSpace = free_space(occupancy);
 73:
 74:     if (bestSpace == -1)
 75:         return false;
 76:     else
 77:     {
 78:         names[bestSpace] = name;
 79:         occupancy[bestSpace] = status;
 80:         return true;
 81:     }
 82: }
 83:
 84: void move_staff(string names[], int occupancy[])
 85: {
 86:     const int STAFF = 1;
 87:
 88:     for (int i = 26; i < NUM_SPOTS; i++)
 89:     {
 90:         if (occupancy[i] == STAFF)
 91:         {
 92:             if (free_space(occupancy) < i)
 93:             {
 94:                 assign_space(names, occupancy, names[i], STAFF);
 95:                 free_up_space(names, occupancy, names[i]);
 96:             }
 97:         }
 98:
 99:     }
100: }
```

```
101:
102: void output_file(ofstream & fout, string names[], int occupancy[])
103: {
104:     for(int i = 1; i < NUM_SPOTS; i++)
105:         if (occupancy[i] != -1)
106:             fout << occupancy[i] << " " << names[i] << " " << i << endl;
107:
108: }
109:
110: int main()
111: {
112:
113:     ifstream fin_current("parking_current.txt");
114:     ifstream fin_add("parking_add.txt");
115:     ifstream fin_remove("parking_remove.txt");
116:     ofstream fout("parking_updated.txt");
117:
118:     if (!fin_current || !fin_add || !fin_remove ||!fout)
119:     {
120:         cout << "Could not open file.";
121:         return EXIT_FAILURE;
122:     }
123:
124:     string names[NUM_SPOTS] = {""};
125:     int occupancy[NUM_SPOTS] = {-1};
126:
127:     //default all parking spots to empty until filled
128:     for (int i = 0; i < NUM_SPOTS; i++)
129:         occupancy[i] = -1;
130:
131:     read_current(fin_current, names, occupancy);
132:
133:     string RAnames[51] = {""};
134:     int RAoccupancy[51] = {0};
135:     int numRemoved = remove_add(fin_remove, RAnames, RAoccupancy);
136:
137:     for (int i = 1; i < numRemoved; i++)
138:         free_up_space(names, occupancy, RAnames[i]);
139:
140:     int numAdded = remove_add(fin_add, RAnames, RAoccupancy);
141:
142:     for(int i = 1; i < numRemoved; i++)
143:         assign_space(names, occupancy, RAnames[i], RAoccupancy[i]);
144:
145:     move_staff(names, occupancy);
146:
147:     output_file(fout, names, occupancy);
148:
149:     fin_current.close();
150:     fin_add.close();
```

```
151:     fin_remove.close();
152:     fout.close();
153:
154: }
155:
```