```cpp
//Kalli Bonin and Ethan Goold
//Question 1 - Parking Services

#include <iostream>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iomanip>

using namespace std;

const int NUM_SPOTS = 51;

void read_current(ifstream & fin, string names[], int occupancy[])
{
    int count = 0;
    int tempOccp = -1;
    string tempName = "";
    int spotNum = 0;

    while (count < NUM_SPOTS && fin >> tempOccp)
    {
        fin >> tempName >> spotNum;

        names[spotNum] = tempName;
        occupancy[spotNum] = tempOccp;
    }
}

int remove_add(ifstream & fin, string RAnames[], int RAoccupancy[])
{
    int index = 0;
    int status = 0;
    string name = "";

    while (fin >> status)
    {
        fin >> name;
        RAoccupancy[index] = status;
        RAnames[index] = name;
        index++;
    }
    return index;
}

void free_up_space(string names[], int occupancy[], string name)
{
    for (int i = 0; i < NUM_SPOTS; i++)
    {
        if (name == names[i])
        {
            names[i] = "";
            occupancy[i] = -1;
        }
    }
}

int free_space(int occupancy[])
{
    for (int i = 1; i < NUM_SPOTS; i++)
    {
        if (occupancy[i] == -1)
            return i;
        //all parking spots are full
        else
            return -1;
    }
}

bool assign_space (string names[], int occupancy[], string name, int status)
{
    int bestSpace = free_space(occupancy);

    if (bestSpace == -1)
        return false;
    else
    {
        names[bestSpace] = name;
        occupancy[bestSpace] = status;
        return true;
    }
}

void move_staff(string names[], int occupancy[])
{
    const int STAFF = 1;

    for (int i = 26; i < NUM_SPOTS; i++)
    {
        if (occupancy[i] == STAFF)
        {
            if (free_space(occupancy) < i)
            {
                assign_space(names, occupancy, names[i], STAFF);
                free_up_space(names, occupancy, names[i]);
            }
        }

    }
}
```

```

void output_file(ofstream & fout, string names[], int occupancy[])
{
    for(int i = 1; i < NUM_SPOTS; i++)
        if (occupancy[i] != -1)
            fout << occupancy[i] << " " << names[i] << " " << i << endl;

}

int main()
{

    ifstream fin_current("parking_current.txt");
    ifstream fin_add("parking_add.txt");
    ifstream fin_remove("parking_remove.txt");
    ofstream fout("parking_updated.txt");

    if (!fin_current || !fin_add || !fin_remove ||!fout)
    {
        cout << "Could not open file.";
        return EXIT_FAILURE;
    }

    string names[NUM_SPOTS] = {""};
    int occupancy[NUM_SPOTS] = {-1};

    //default all parking spots to empty until filled
    for (int i = 0; i < NUM_SPOTS; i++)
        occupancy[i] = -1;

    read_current(fin_current, names, occupancy);

    string RAnames[51] = {""};
    int RAoccupancy[51] = {0};
    int numRemoved = remove_add(fin_remove, RAnames, RAoccupancy);

    for (int i = 1; i < numRemoved; i++)
        free_up_space(names, occupancy, RAnames[i]);

    int numAdded = remove_add(fin_add, RAnames, RAoccupancy);

    for(int i = 1; i < numRemoved; i++)
        assign_space(names, occupancy, RAnames[i], RAoccupancy[i]);

    move_staff(names, occupancy);

    output_file(fout, names, occupancy);

    fin_current.close();
    fin_add.close();
    fin_remove.close();
    fout.close();

}

```