Code:

```
#include <bits/stdc++.h>
#include <vector>
#include <queue>
#include <unordered_set>

using namespace std;

bool isValidMove(int current, int next, const vector<int>& blockedCells) {
    if (next < 0 || next >= 25 || find(blockedCells.begin(), blockedCells.end(), next) != blockedCells.end()) {
        return false;
    }

    if (current % 5 == 0 && next % 5 == 4) {
        return false;  // Moving left from the leftmost column
    } else if (current % 5 == 4 && next % 5 == 0) {
        return false;  // Moving right from the rightmost column
    }

    return true;
}

vector<vector<int>> findAllShortestRoutes(const vector<int>& blockedCells) {
    int start = 0, end = 24;

    queue<pair<int, vector<int>>> q;
    q.push({start, {start}});

    unordered_set<int> visited;
```

```
vector<vector<int>> shortestRoutes;

while (!q.empty()) {
    int current = q.front().first;
    vector<int> path = q.front().second;
    q.pop();

    if (current == end) {
        shortestRoutes.push_back(path);
        continue;
    }

    if (visited.count(current)) {
        continue;  // Skip already visited nodes
    }

    visited.insert(current);

    vector<int> neighbors = {
        current - 1,  // Move left
        current + 1,  // Move right
        current - 5,  // Move up
        current + 5   // Move down
    };

    for (int neighbor : neighbors) {
        if (isValidMove(current, neighbor, blockedCells)) {
            vector<int> newPath = path;
            newPath.push_back(neighbor);
            q.push({neighbor, newPath});
        }
```

```
        }
    }

    return shortestRoutes;
}

int main() {
    vector<int> blockedCells = {7,9,10,14,17,18,19};
    vector<vector<int>> shortestRoutes = findAllShortestRoutes(blockedCells);

    cout << "All Shortest Routes:" << endl;
    for (const auto& route : shortestRoutes) {
        for (int cell : route) {
            cout << cell << " ";
        }
        cout << endl;
    }

    return 0;
}
```