```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.


#include <vector>
#include <unordered_map>
#include <climits>
#include <algorithm>
#include <cmath>

using namespace std;

vector<int> getMinDistances(vector<unordered_map<string, bool>> blocks,
                            string req);
vector<int> getMaxDistancesAtBlocks(vector<unordered_map<string, bool>> blocks,
                                    vector<vector<int>> minDistancesFromBlocks);
int getIdxAtMinValue(vector<int> array);
int distanceBetween(int a, int b);

// O(br) time | O(br) space - where b is the number of blocks and r is the
// number of requirements
int apartmentHunting(vector<unordered_map<string, bool>> blocks,
                     vector<string> reqs) {
  vector<vector<int>> minDistancesFromBlocks;
  for (string req : reqs) {
    minDistancesFromBlocks.push_back(getMinDistances(blocks, req));
  }
  vector<int> maxDistancesAtBlocks =
      getMaxDistancesAtBlocks(blocks, minDistancesFromBlocks);
  return getIdxAtMinValue(maxDistancesAtBlocks);
}

vector<int> getMinDistances(vector<unordered_map<string, bool>> blocks,
                            string req) {
  vector<int> minDistances(blocks.size());
  int closestReqIdx = INT_MAX;
  for (int i = 0; i < blocks.size(); i++) {
    if (blocks[i][req])
      closestReqIdx = i;
    minDistances[i] = distanceBetween(i, closestReqIdx);
  }
  for (int i = blocks.size() - 1; i >= 0; i--) {
    if (blocks[i][req])
      closestReqIdx = i;
    minDistances[i] = min(minDistances[i], distanceBetween(i, closestReqIdx));
  }
  return minDistances;
}

vector<int>
getMaxDistancesAtBlocks(vector<unordered_map<string, bool>> blocks,
                        vector<vector<int>> minDistancesFromBlocks) {
  vector<int> maxDistancesAtBlocks(blocks.size());
  for (int i = 0; i < blocks.size(); i++) {
    vector<int> minDistancesAtBlock;
    for (vector<int> distances : minDistancesFromBlocks) {
      minDistancesAtBlock.push_back(distances[i]);
    }
    maxDistancesAtBlocks[i] =
        *max_element(minDistancesAtBlock.begin(), minDistancesAtBlock.end());
  }
  return maxDistancesAtBlocks;
}

int getIdxAtMinValue(vector<int> array) {
  int idxAtMinValue = 0;
  int minValue = INT_MAX;
  for (int i = 0; i < array.size(); i++) {
    int currentValue = array[i];
    if (currentValue < minValue) {
      minValue = currentValue;
      idxAtMinValue = i;
    }
  }
  return idxAtMinValue;
}

int distanceBetween(int a, int b) { return abs(a - b); }

```