```
#include <iostream>
#include <vector>
using namespace std;

int** makeAMove(int** board, int x, int y, int player,int isCheck);

vector<pair<int, int>*>* possibleMoves(int** board, int player) {
 vector<pair<int, int>*>* output = new vector<pair<int, int>*>();
 for (int i = 0; i < 4; i++) {
  for (int j = 0; j < 4; j++) {
   if (board[i][j] == 0) {
    if (makeAMove(board, i, j, player, true)) {
     pair<int, int>* p = new pair<int, int>(i, j);
     output->push_back(p);
    }
   }
  } }
 return output;
}

int evaluateBoard(int** board, bool& completed) {
 if (possibleMoves(board, 1)->size() != 0 || possibleMoves(board, 2)->size() != 0) {
  completed = false;
  return 0;
 }
 // cout << "here" << endl;
 completed = true;
 int count[3] = {};
 for (int i = 0; i < 4; i++) {
  for (int j = 0; j < 4; j++) {
   count[board[i][j]]++;
  }
 }
 return count[1] - count[2];
}

int** makeAMove(int** board, int x, int y, int player,int isCheck) {

 int** newBoard;

 if(isCheck){
  newBoard = board;
 }else{
  newBoard = new int*[4];
  for (int i = 0; i < 4; i++) {
   newBoard[i] = new int[4];
   for (int j = 0; j < 4; j++) {
    newBoard[i][j] = board[i][j];
   }
  }
 }

 int xDir[] = {-1,-1,0,1,1,1,0,-1};
 int yDir[] = {0,1,1,1,0,-1,-1,-1};
 if(x < 0 || x >= 4 || y < 0 || y >= 4 || board[x][y] != 0){
  return NULL;
 }
 bool movePossible = false;
 for(int i = 0; i < 8; i++){
  int xStep = xDir[i];
  int yStep = yDir[i];
  int currentX = x + xStep;
  int currentY = y + yStep;
  int count = 0;
  while(currentX >= 0 && currentX < 4 && currentY >= 0 && currentY < 4){

   if(newBoard[currentX][currentY] == 0){
    break;
   }else if(newBoard[currentX][currentY] != player){
    currentX += xStep;
    currentY += yStep;
    count++;
   }else{
    if(count > 0){
     movePossible = true;
     if(isCheck){
      return newBoard;
     }
     int convertX = currentX - xStep;
     int convertY = currentY - yStep;
     while(convertX != x || convertY != y){
      newBoard[convertX][convertY] = player;
      convertX = convertX - xStep;
      convertY = convertY - yStep;
     }
     newBoard[x][y] = player;
    }
    break;
   }
  }
 }
 if(movePossible){
  return newBoard;
 }else{
  return NULL;
 }
}

int kkk = 0;

void printBoard(int** board) {
 for (int i = 0 ; i < 4; i++) {
  for(int j = 0; j < 4; j++) {
   cout << board[i][j] << " ";
  }
  cout << endl;
 }
}

int findNextStep(int** board, bool maximizerTurn, int& x, int& y) {
 // check if done with play if yes return score of evaluate function
 bool completed;
 int score = evaluateBoard(board, completed);
 printBoard(board);
 if (completed) {
  return score;
 }

 // find all possible moves
 int player = maximizerTurn? 1 : 2;
 vector<pair<int, int>*>* options = possibleMoves(board, player);
 if (options->size() == 0) {
  x = -1;
  y = -1;
  int dummyX, dummyY;
  return findNextStep(board, !maximizerTurn, dummyX, dummyY);
 }
 int best;
 if (maximizerTurn) {
  best = INT_MIN;
 } else {
  best = INT_MAX;
 }
 // for each move
 for (int i = 0; i < options->size(); i++) {
  //  Make the move
  int currentMoveX = options->at(i)->first;
  int currentMoveY = options->at(i)->second;
  int** newBoard = makeAMove(board, currentMoveX, currentMoveY, player, false);
  // Make recursive call
  int dummyX, dummyY;

  int score = findNextStep(newBoard, !maximizerTurn, dummyX, dummyY);
  //  update best (will be min/max depending on who's turn it is)
  if (maximizerTurn) {
   best = max(best, score);
  } else {
   best = min(best, score);
  }
  if (best == score) {
   x = currentMoveX;
   y = currentMoveY;
  }
  // Undo the move. We will have to maintain a copy of the board for this.
  for (int i = 0; i < 4; i++) {
   delete [] newBoard[i];
  }
  delete [] newBoard;
 }
 // update x & y and return best score
 return best;
}

int main() {
 int **a = new int*[4];
 for (int i = 0; i < 4; i++) {
  a[i] = new int[4];
  for (int j = 0; j < 4; j++) {
   a[i][j] = 0;
  }
 }
 a[1][1] = 1;
 a[2][2] = 1;
 a[1][2] = 2;
 a[2][1] = 2;
 int x, y;
 cout << findNextStep(a, true, x, y) << endl;
 cout << x << " " << y << endl;
}
```