/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Online C++ Compiler.

Code, Compile, Run and Debug C++ program online.

Write your code in this editor and press "Run" button to compile and execute it.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Online C++ compiler to run C++ program online

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <stdexcept>

#include <string>

using namespace std;

int boardSize;

class Ship {

public:

int size;

int hits;

Ship(int size) {

this->size = size;

this->hits = 0;

}

bool isSunk() {

return hits == size;

}

};

class Board {

public:

vector<vector<char>> grid;

vector<Ship> ships;

Board() {

grid.resize(boardSize, vector<char>(boardSize, '-'));

}

void print() {

for (int i = 0; i < boardSize; i++) {

for (int j = 0; j < boardSize; j++) {

if (grid[i][j] == 'S') {

cout << "P ";

} else {

cout << grid[i][j] << " ";

}

}

cout << endl;

}

}

bool isHit(int row, int col) {

for (int i = 0; i < ships.size(); i++) {

for (int j = 0; j < ships[i].size; j++) {

if (ships[i].isSunk()) {

continue;

}

if (i == row && j == col ) {

ships[i].hits++;

grid[i][j] = 'P';

return true;

}

}

}

grid[row][col] = 'P';

return false;

}

bool allShipsSunk() {

for (int i = 0; i < ships.size(); i++) {

if (!ships[i].isSunk()) {

return false;

}

}

return true;

}

void markOccupied(int row, int col, int dir, int size, vector<vector<bool>>& occupied) {

for (int i = 0; i < size; i++) {

if (dir == 0) {

occupied[row+i][col] = true;

} else {

occupied[row][col+i] = true;

}

}

}

void placePlayerShips(Board& playerBoard, int boardSize) {

vector<int> sizes = {5, 4, 3, 3, 2};

vector<vector<bool>> occupiedPlayer(boardSize, vector<bool>(boardSize, false));

if (boardSize>=6 || boardSize<=7){

for (int i =2 ; i < sizes.size(); i++) {

int size = sizes[i];

int row, col, dir;

bool validPlacement = false;

while (!validPlacement) {

playerBoard.print();

cout << "Placing ship of size " << size << endl;

while (true) {

cout << "Enter row (0 - " << boardSize-1 << "): ";

cin >> row;

if (!cin.good()) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter an integer value." << endl;

continue;

}

if (row < 0 || row > boardSize-1) {

cout << "Invalid input. Please enter values between 0 and " << boardSize-1 << "." << endl;

continue;

}

break;

}

while (true) {

cout << "Enter column (0 - " << boardSize-1 << "): ";

cin >> col;

if (!cin.good()) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter an integer value." << endl;

continue;

}

if (col < 0 || col > boardSize-1) {

cout << "Invalid input. Please enter values between 0 and " << boardSize-1 << "." << endl;

continue;

}

break;

}

while (true) {

cout << "Enter direction (0 - vertical, 1 - horizontal): ";

cin >> dir;

if (!cin.good()) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter an integer value." << endl;

continue;

}

if (dir != 0 && dir != 1) {

cout << "Invalid input. Please enter either 0 or 1." << endl;

continue;

}

break;

}

if (isValidPlacement(row, col, dir, size, occupiedPlayer)) {

placeShip(row, col, dir, size);

markOccupied(row, col, dir, size, occupiedPlayer);

break;

}

if (playerBoard.isValidPlacement(row, col, dir, size,occupiedPlayer )) {

playerBoard.placeShip(row, col, dir, size);

validPlacement = true;

} else {

cout << "Invalid placement. Please try again." << endl;

}

}

}

}

if (boardSize==10){

for (int i =0 ; i < sizes.size(); i++) {

int size = sizes[i];

int row, col, dir;

bool validPlacement = false;

while (!validPlacement) {

playerBoard.print();

cout << "Placing ship of size " << size << endl;

while (true) {

cout << "Enter row (0 - " << boardSize-1 << "): ";

cin >> row;

if (!cin.good()) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter an integer value." << endl;

continue;

}

if (row < 0 || row > boardSize-1) {

cout << "Invalid input. Please enter values between 0 and " << boardSize-1 << "." << endl;

continue;

}

break;

}

while (true) {

cout << "Enter column (0 - " << boardSize-1 << "): ";

cin >> col;

if (!cin.good()) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter an integer value." << endl;

continue;

}

if (col < 0 || col > boardSize-1) {

cout << "Invalid input. Please enter values between 0 and " << boardSize-1 << "." << endl;

continue;

}

break;

}

while (true) {

cout << "Enter direction (0 - vertical, 1 - horizontal): ";

cin >> dir;

if (!cin.good()) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter an integer value." << endl;

continue;

}

if (dir != 0 && dir != 1) {

cout << "Invalid input. Please enter either 0 or 1." << endl;

continue;

}

break;

}

}

if (isValidPlacement(row, col, dir, size, occupiedPlayer)) {

placeShip(row, col, dir, size);

markOccupied(row, col, dir, size, occupiedPlayer);

break;

}

if (playerBoard.isValidPlacement(row, col, dir, size,occupiedPlayer )) {

playerBoard.placeShip(row, col, dir, size);

validPlacement = true;

} else {

cout << "Invalid placement. Please try again." << endl;

}

}

}

}

void placeShips() {

vector<int> sizes = {5, 4, 3, 3, 2};

vector<vector<bool>> occupied(boardSize, vector<bool>(boardSize, false));

if(boardSize>6 || boardSize<8){

for (int i = 2; i < sizes.size(); i++) {

int size = sizes[i];

int row, col, dir;

int attempts = 0; // зберігає кількість спроб розстановки корабля

while (attempts < 100) {

row = rand() % boardSize;

col = rand() % boardSize;

dir = rand() % 2;

if (isValidPlacement(row, col, dir, size, occupied)) {

placeShip(row, col, dir, size);

markOccupied(row, col, dir, size, occupied); // позначаємо зайняті клітинки

break;

}

attempts++;

}

}

}else if (boardSize=10){

for (int i = 0; i < sizes.size(); i++) {

int size = sizes[i];

int row, col, dir;

int attempts = 0; // зберігає кількість спроб розстановки корабля

while (attempts < 100) {

row = rand() % boardSize;

col = rand() % boardSize;

dir = rand() % 2;

if (isValidPlacement(row, col, dir, size, occupied)) {

placeShip(row, col, dir, size);

markOccupied(row, col, dir, size, occupied); // позначаємо зайняті клітинки

break;

}

attempts++;

}

}

}

}

bool isValidPlacement(int row, int col, int dir, int size, const vector<vector<bool>>& occupied) {

if ((dir == 0 && row + size > boardSize) || (dir == 1 && col + size > boardSize)) {

return false;

}

// Перевірка чи клітини вже зайняті кораблями

for (int i = 0; i < size - 1; i++) {

if (dir == 0 && occupied[row + i][col]) {

return false;

}

if (dir == 1 && occupied[row][col + i]) {

return false;

}

}

// Перевірка чи остання клітини корабля вільна

int lastRow = row + (size - 1) \* dir;

int lastCol = col + (size - 1) \* (1 - dir);

if (occupied[lastRow][lastCol]) {

return false;

}

// Перевірка чи клітини навколо корабля вільні

for (int r = row - 1; r <= lastRow + 2; r++) {

for (int c = col - 1; c <= lastCol + 2; c++) {

if (r >= 0 && r < boardSize && c >= 0 && c < boardSize) {

if (occupied[r][c]) {

return false;

}

}

}

}

return true;

}

private:

bool isValidPlacement(int row, int col, int dir, int size) {

if (dir != 0 && dir != 1) {

throw invalid\_argument("Invalid direction.");

}

if (size < 1 || size > boardSize) {

throw out\_of\_range("Invalid ship size.");

}

if (dir == 0 && row + size > boardSize) {

return false;

} else if (dir == 1 && col + size > boardSize) {

return false;

}

return true;

}

void placeShip(int row, int col, int dir, int size) {

Ship ship(size);

ships.push\_back(ship);

for (int i = 0; i < size; i++) {

if (dir == 0) {

grid[row + i][col] = 'X';

} else {

grid[row][col + i] = 'X';

}

}

}

};

class Game {

public:

Board playerBoard;

Board computerBoard;

Board computerBoardS;

Game() {

srand(time(NULL));

playerBoard.placePlayerShips(playerBoard,boardSize);

// playerBoard.placeShips();

computerBoard.placeShips();

}

void play() {

cout << "Welcome to Battleship!" << endl;

while (true) {

cout << "Your board:" << endl;

playerBoard.print();

cout << "Computer's board:" << endl;

// computerBoard.print();

computerBoardS.print();

cout << "Enter row and column to attack count of rows and colums starts from 0 (e.g. 3 4): ";

int row, col;

while (!(cin >> row >> col) || row < 0 || row >= boardSize || col < 0 || col >= boardSize) {

cin.clear();

cin.ignore();

cout << "Invalid input. Please enter a row and column within the range (0-" << boardSize - 1 << "): ";

}

if (computerBoardS.isHit(row, col) == true || computerBoard.isHit(row, col) == true) {

cout << "You hit a ship!" << endl;

} else {

cout << "You missed." << endl;

}

if (computerBoard.allShipsSunk()) {

cout << "You win!" << endl;

break;

}

row = rand() % boardSize;

col = rand() % boardSize;

if (playerBoard.isHit(row, col)) {

cout << "Computer hit one of your ships!" << endl;

} else {

cout << "Computer missed." << endl;

}

if (playerBoard.allShipsSunk()) {

cout << "Computer wins!" << endl;

break;

}

}

}

};

int main() {

int rules;

cout << "Want to read rules? (yes - 1 no - 0) ";

cin >> rules;

if(rules == 1){

cout<< "If you have board 6-7 you will have 3 ships. "<<endl;

cout<< "If you have board 8-10 you will have 5 ships. "<<endl;

}

while (rules!=1 && rules!=0 || cin.fail() ) {

cin.clear(); // скидаємо флаги помилок

cin.ignore(); // очищення введення

cout << "Invalid input. Please enter a valid number: ";

cin >> rules;

}

cout << "Input board size from 6 to 10: ";

cin >> boardSize;

while (boardSize < 6 || boardSize > 10 || cin.fail()) {

cin.clear(); // скидаємо флаги помилок

cin.ignore(); // очищення введення

cout << "Invalid input. Please enter a valid number: ";

cin >> boardSize;

}

cout << "Board size is " << boardSize << endl;

Game game;

game.play();

return 0;

}