Battleship

Project2

CSC- 5 – 46023 Intro C++

Tsz,Kwan

28 – July – 2014

Content

1.Introduction………………………………………….3  
Rule and Gameplay  
Thoughts after Program

2. Development………………………..………………4-5  
Approach Strategy

3. Variables list…………………………….……..……5-6

4. Topic Covered (Checklist)…………………….……6-7

5. Libraries included……………………………………..7

6. Pseudo Code……………………………………….8-10

7. Flowchart…………………………………………11-15

8. Code……………………………………………...16-46

1. Introduction

**Rules and Gameplay**

There are 2 10x10 tables shown on the screen. One is for player, the other is for AI. Both of them have 5 ships which are 1 5-unit ship, 1 4-unit ship,1 3-unit ship, and 2 2-unit ships. After the player enter the coordinate to place the ship, the table will be refresh and use 2 to 5 to label the ships. The coordinates of AI’s ships will be place randomly. Player needs to hit all the ships to win the game. In the game, X means hit and O means miss. Every time after check the validation, the program will scan the table to check whether there is any numbers which is ships on the table. If there isn’t any numbers on the table, the game ends.

**Thoughts after Program**

The game seems very simple, but the AI’s fire part is very complicated because it is very difficult to make an AI act like a human player. I want the AI check the coordinates around the hit coordinate, and keep fire when he gets the second hit. If one side is “O” or the side of the table, the AI needs to check the other side too. This needs many Boolean variables. Also, the AI table shown to the player isn’t the real table, it is a clear table and after the player fire, the program will compare the coordinate to the real table. Then record and show “O”, “X”, or invalid input. The validation part costs me a lot of time too because I use string as an input type and input in A1 form to let the player input the coordinates. This can check the length easily, but I need to use ascii code to translate after check the length. It is possible to make the Ai smarter which is divided the table into several sections and randomly fire each of the section to increase the accuracy, but it needs more codes and better logic.

2. Development

Approach Strategy

The battleship needs three 10x10 tables, it is too difficult to use one-dimension arrays. It is easier to use 2-dimension arrays. Also, I use A-J to label the rows and 0-9 to label the columns. It makes the players enter the coordinates clearly and prevent them get confused. I have tried to let the player to choose which ship they want to place first, but there are 2 2-unit ships, so I need to use a Boolean to remember the first 2-unit ship. However, there are many bugs and I couldn’t fix it. Therefore, I let the player place the 5-unit ship, then 4-unit ship, and so on. After the player’s place ship part, I need to random the AI ships’ coordinates. Because I use an array to store the ship units, so I can avoid the oversize by subtract the units such as srand()%10-5. After generate the coordinate, the program will random to place it horizontally or vertically. If the ship overlaps, it will try to place it in other way. If it is still invalid, the program will random the coordinates again.

After the preparing, I use a switch to separate the player’s fire turn and AI’s fire turn. If the game isn’t over, the program to go to AI’s turn and so on. If the game ends in player’s turn, the program will jump to other case same as AIs. Also, I put a do-while loop outside the switch and repeat until the game is over.

For the AI’s fire part, I let the AI to fire randomly until it hits. After AI hits, the program will record the coordinate and check the four coordinates beside it until it gets second hit. After a second hit, AI will fire that direction until it get miss, touch the side, or overlap. Then, it will fire the opposite side until miss, oversize, overlap again. After it finishes these steps, it will go back to random fire mode.

3. Variables list

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Line** |
| int | COLS=10 | Global const | 19 |
|  | COL=3 | Global const | 20 |
|  | ROWS=10 | const | 34 |
|  | XY=4 | const | 34 |
|  | ROW=100 | const | 34 |
|  | unit[5] ={5,4,3,2,2} | unit of ship | 38 |
|  | x1,y1,x2,y2 | coordinate to place ship | 39 |
|  | hx=10, hy=10 | first hit coordinates | 43 |
|  | ax,ay | ai fire coordinate | 61 |
|  | hplan | hit plan after first hit (corss) | 64 |
|  | oppcombo=0 | the other side | 46 |
|  | turn | switch turn | 48 |
|  | count | use space check validation | 494 |
|  | max, min | replace the coordinates to place ship | 495 |
| float | phit=0, pmiss=0 | player hit miss counter | 53 |
|  | aihit=0, aimiss=0 | ai hit miss counter | 54 |
| char | p[ROWS][COLS] | player table | 35 |
|  | ai[ROWS][COLS] | ai fake table | 36 |
|  | real[ROWS][COLS] | real ai table | 37 |
|  | pvect[ROW][COL] | player vector array to do sorting | 58 |
|  | aivect[ROW][COL] | ai vector array to do sorting | 59 |
|  | cax,cay | ai fire display in A0 form to player | 62 |
|  | row=i+65 | display A-J | 471 |
|  | temp | temporary memory | 736 |
| string | place | x,y to place ship start and end coordinates | 496 |
|  | fire | player fire | 645 |
| bool | goback=ture | invalid back to random | 40 |
|  | valid | check validation | 41 |
|  | hit | hit to skip random fire | 44 |
|  | finish=true | finish one ship back to random | 45 |
|  | over | game over | 47 |
|  | oneend=false | one side miss/overalp/overside go to opposite direction | 49 |
|  | cross[XY]={true,true,true,true} | cross 4 boxes around hit | 50 |
|  | crossdone=true | if true back to random | 51 |
|  | combohit | keep fire the same direction | 52 |
|  | done | finish fire | 60 |
|  | swap | sorting swap | 735 |
| vector<int> | prow,pcol,airow,aicol | player/ai hit/miss coordinates | 56 |
| vector<char> | pr, air | player/ai hit/miss result | 57 |
| ofstream | output |  | 55 |
| time\_t | start, end | delay display ai fire | 485 |

4. Topic Covered (Checklist)

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter | type | code | line |
| 2.1 Variables | int | int x1,y1,x2,y2; | 39 |
| 2.2 Input Output | cin | cin>>place; | 506 |
|  | cout | cout<<"Hit!!!\n"; | 672 |
|  | endl | cout<<endl; | 723 |
| 2.3 data types | char | char cax,cay; | 431 |
|  | float | float phit=0, pmiss=0; | 53 |
|  | bool | bool hit; | 44 |
|  | string | string place; | 496 |
| 2.4 condition | = | int hx=10, hy=10; | 43 |
|  | == | if(y1==y2){ | 524 |
|  | ++ | count++; | 543 |
| 2.5 style | comment | //player table y,x | 35 |
| 3.1 boolean expression | >=, &&, <= | if(real[y1][x1]>='2' && real[y1][x1]<='5'){ | 671 |
|  | <, >, || | if(ay<0 || ay>9 || ax<0 || ax>9){ | 157 |
| 3.2 multiway branches | switch | switch(turn){ | 89 |
|  | if | if(fire.length()!=2){ | 655 |
|  | else | else{ | 677 |
|  | else if | else if(p[ay][ax]=='X' || p[ay][ax]=='O'){ | 161 |
|  | nested | for(int q=0;q<5;q++){ | 498 |
|  |  | do{ | 499 |
|  | break | break; | 94 |
| 3.3 type of loop | for | for(int i=0;i<4;i++){ | 101 |
|  | do-while | do{}while(valid==false); | 110,118 |
| 4.2 predefined function | srand, time | srand(static\_cast<unsigned int>(time(0))); | 67 |
|  | rand | y1=rand()%(10-unit[q]); | 610 |
| 5.1 void function | void | void intro(); | 22 |
| 5.2 call-by-reference |  | void aiplace(char [][COLS], char [][COLS], int &, int &, bool &, int []); | 25 |
| 6.1 streams and basic | ofsream declare | ofstream output; | 55 |
|  | output | output.open("stat.dat") | 433 |
|  | close | output.close(); | 454 |
| 7.1 array | int array | int unit[5]={5,4,3,2,2}; | 38 |
|  | bool aray | bool cross[XY]={true,true,true,true}; | 50 |
| 7.2 array in function | pass 2d array to function | void table(char [][COLS],char [][COLS], char [][COLS]); | 23 |
| 7.3 soritng | sorting |  | 734-817 |
| 7.4 mutli-dim array | 2D | char p[ROWS][COLS]; | 35 |
| 8.3 vector | int | vector<int> prow,pcol,airow,aicol; | 56 |
|  | char | vector<char> pr, air; | 57 |
|  | pass vector to function by ref | void sort(char [][COL], char [][COL], vector<char>, vector<char>); | 29 |
| difftime |  | }while(difftime(end,start)<1); | 490 |

5. Libraries included

* <cstdlib>
* <iostream>
* <ctime>
* <fstream>
* <vector>
* <iomanip>

6. Pseudo Code

Initialize

Reset table

Output table

do{

Input 2 coordinates to place ship

}while (invalid)

place other ship and check validation

do{

AI random ship coordinates

}while (invalid)

case1

Player enter coordinate to frie

check validation

check hit/miss and add count

display table again

check game over (no number s on the table)

if(true) case3

else case2

case2 (AI fire)

do{

if (not hit/combo) random hit

if (hit) check cross 4

if(all invalid) go back to random

if(hit) combo++, add count

else add count

if (cross 4 coordinates hit) continue fire that direction

if(invalid) jump to next statement, oppcombo++, combo=0

if(miss) oppcombo++, combo=0, add count

if(hit) combo++, add count

if(oppcombo>0) check the opposite side

if(invalid) go back to random

if(miss) oppcombo=0, add count

if(hit) oppcombo+1

}while (not fire)

check game over

if (true) go to case 4

else go to case1

case3

Player win

case 4

Player lose

if(not case 3 && not case4) keep looping the case

scan player and ai table

push back X and O coordinates

copy to player and ai 2d array

sorting

array like A0 X

X(hit) first, O(miss) after

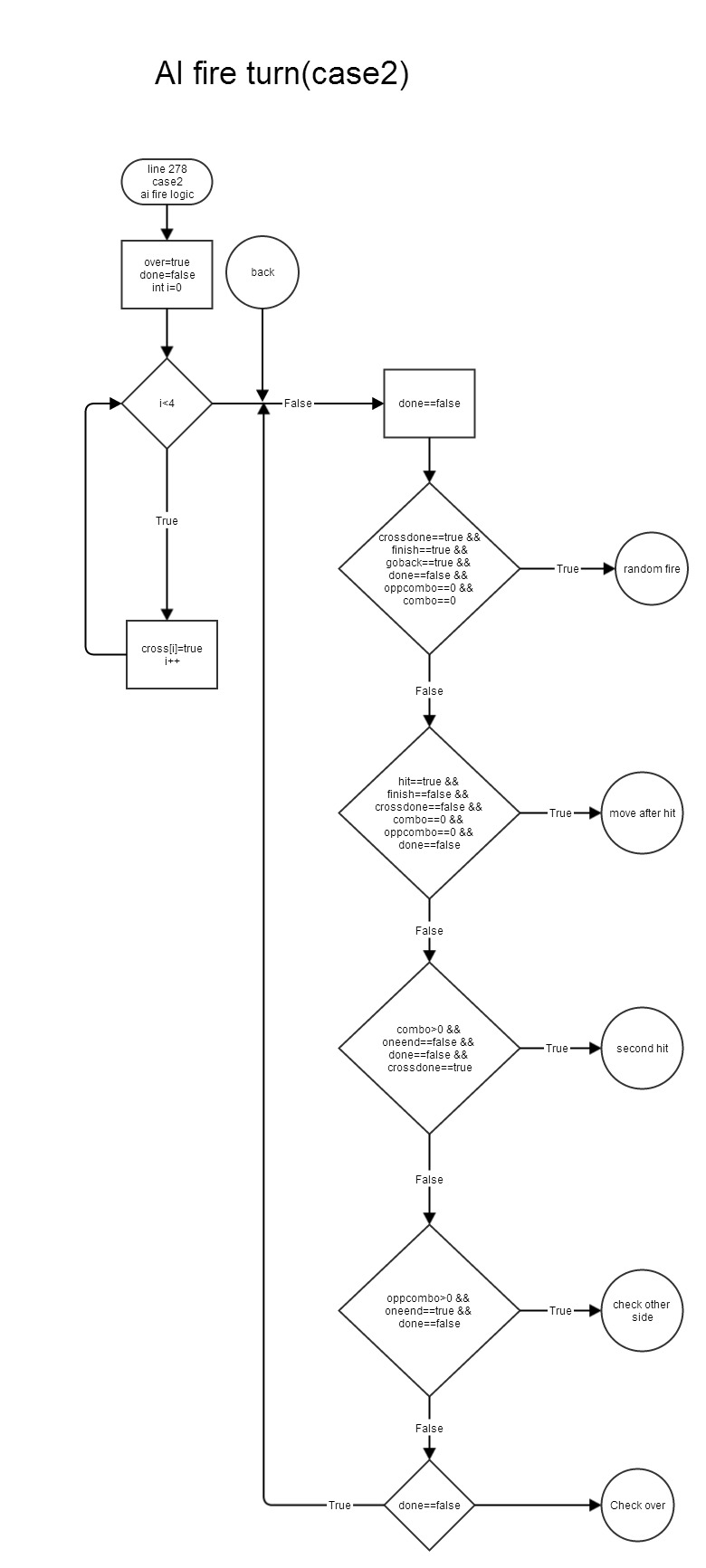
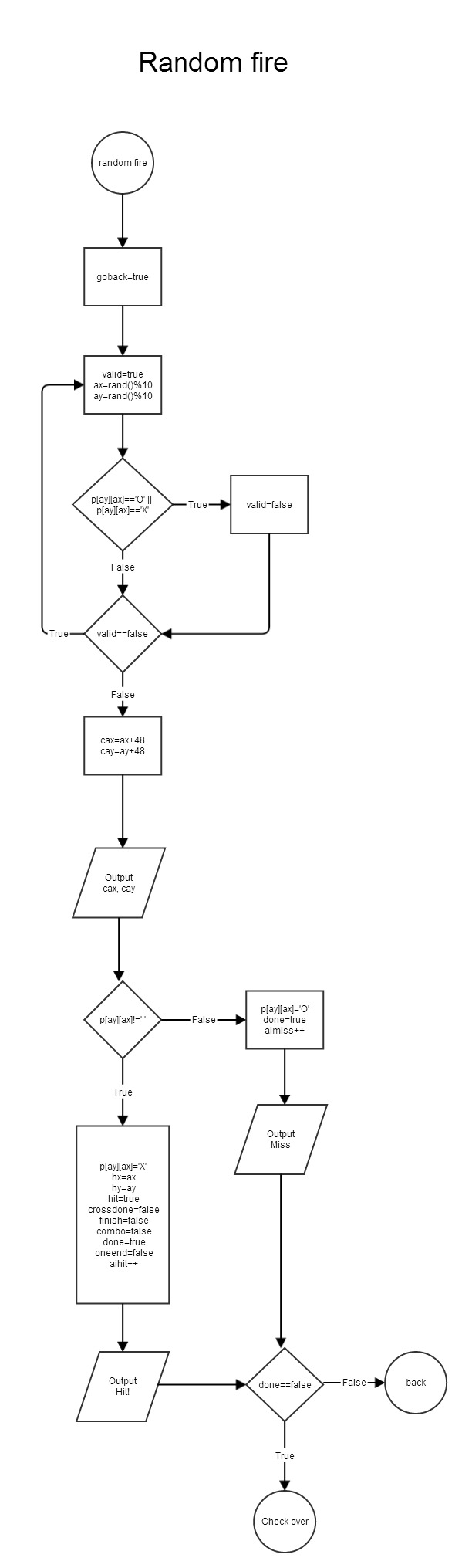
sort with first column

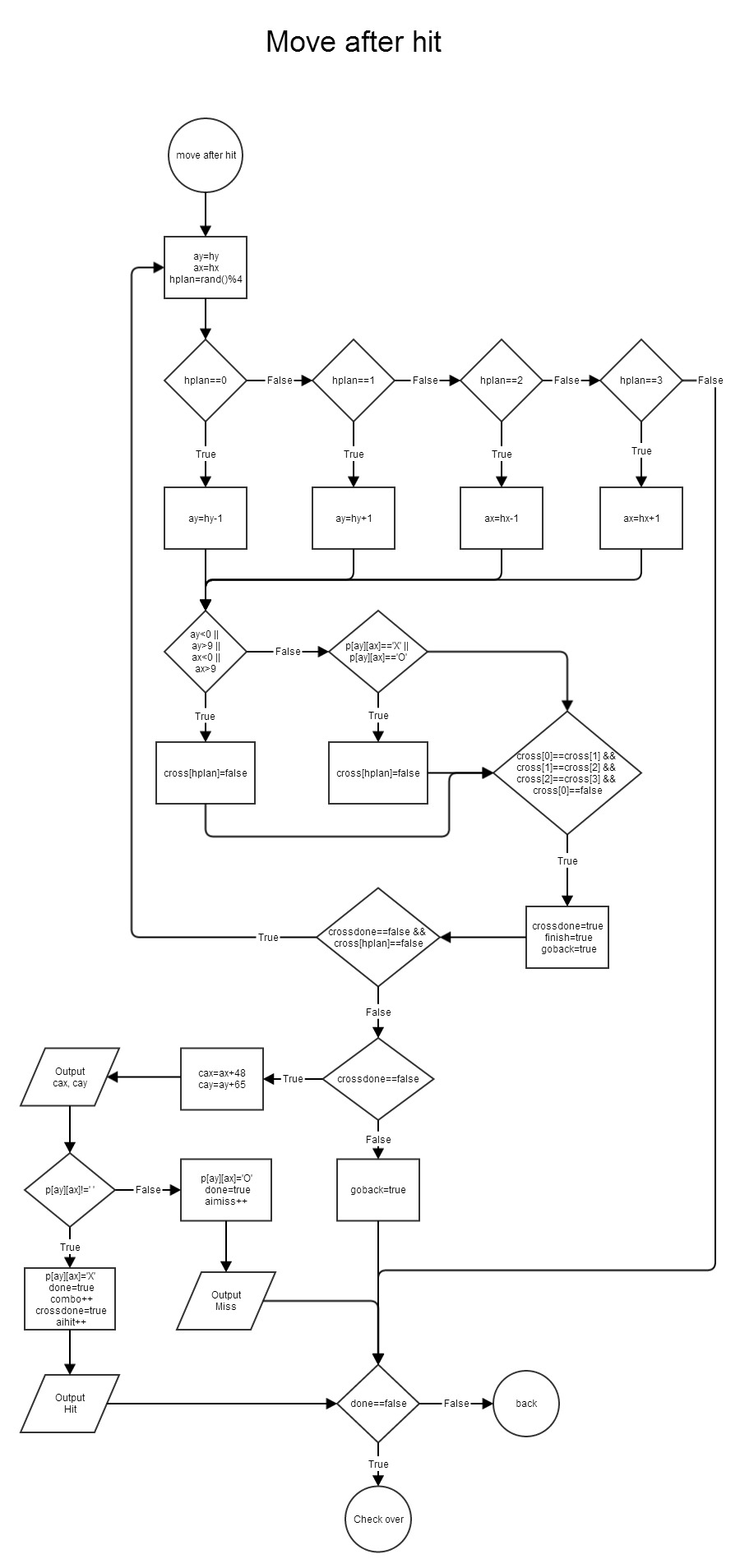
sort with second column

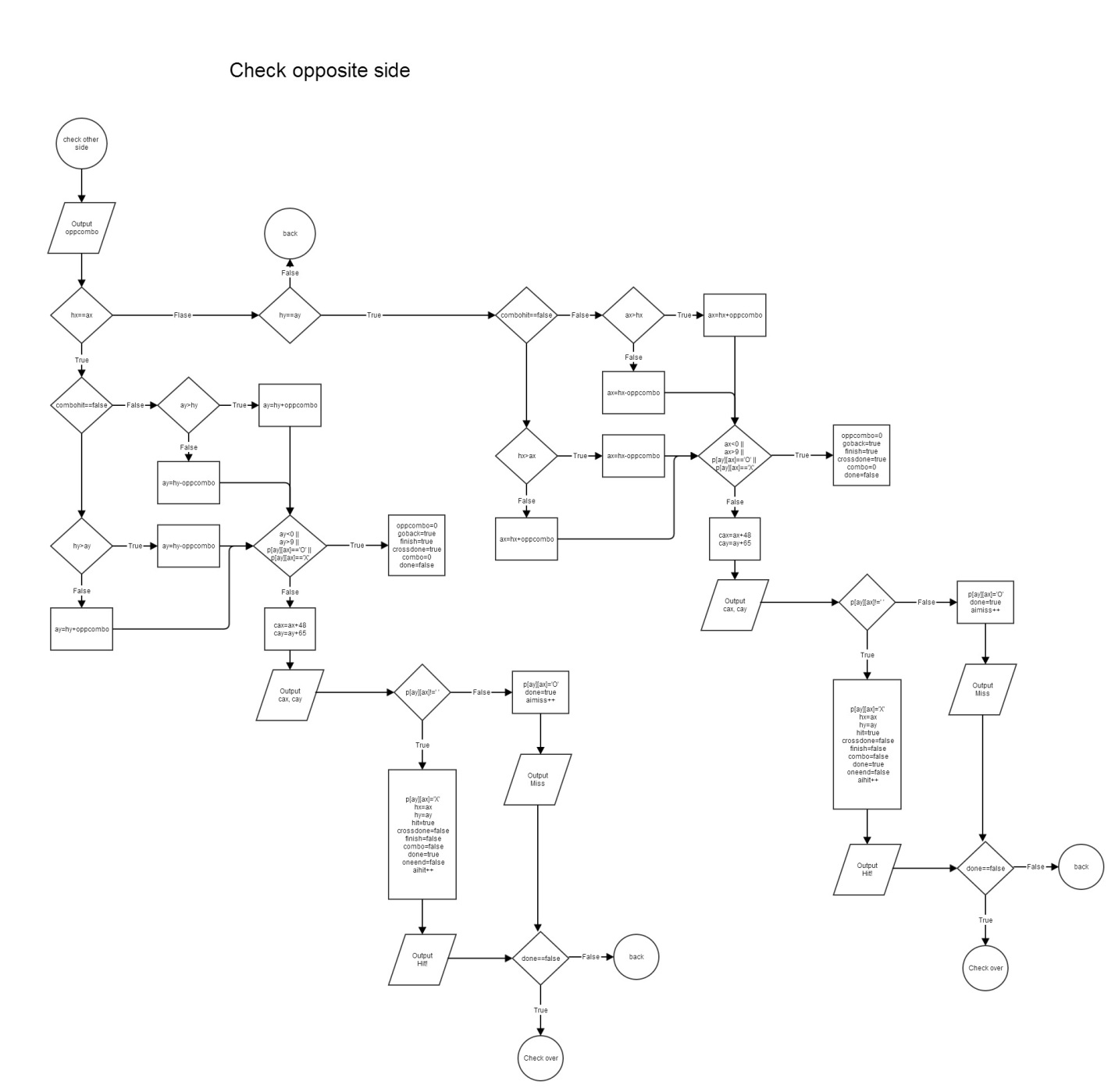
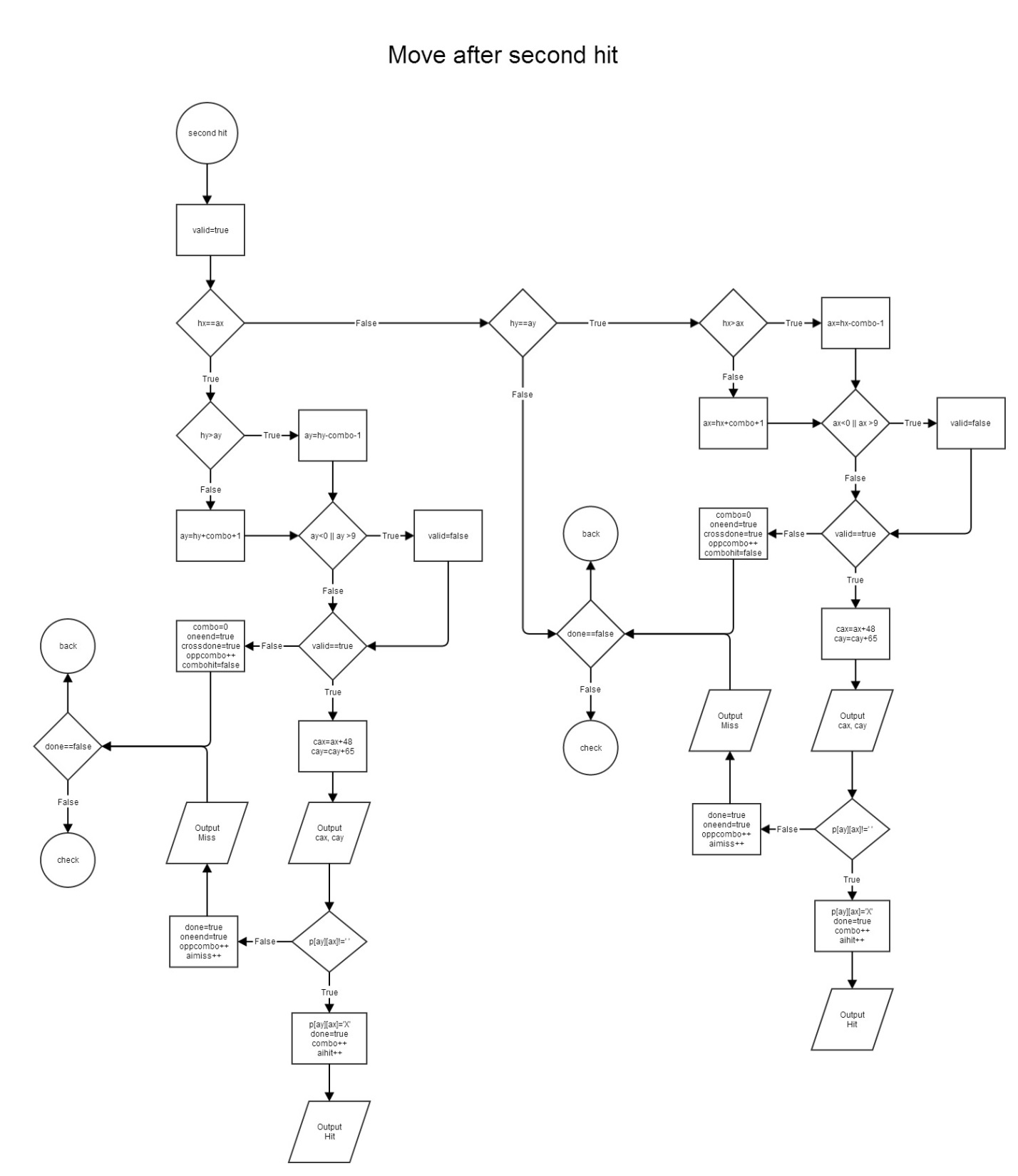
calculate accuracy hit/(hit+miss)

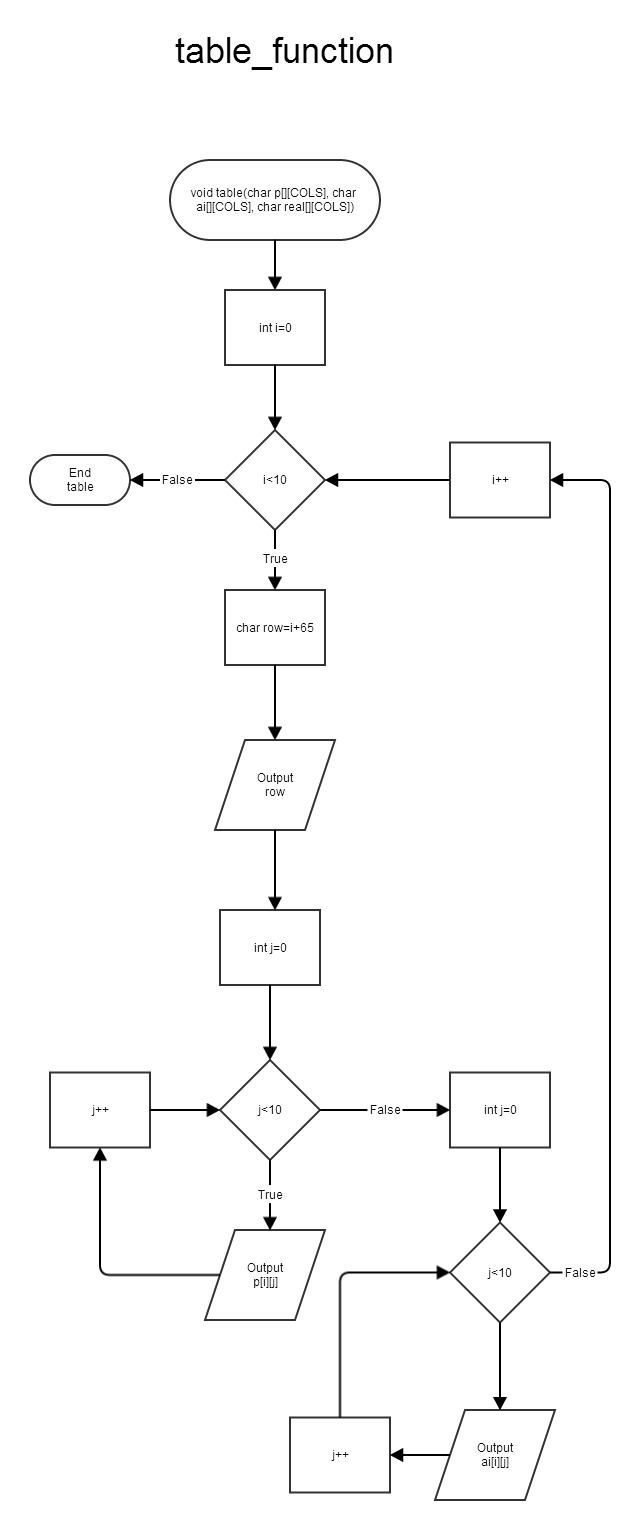
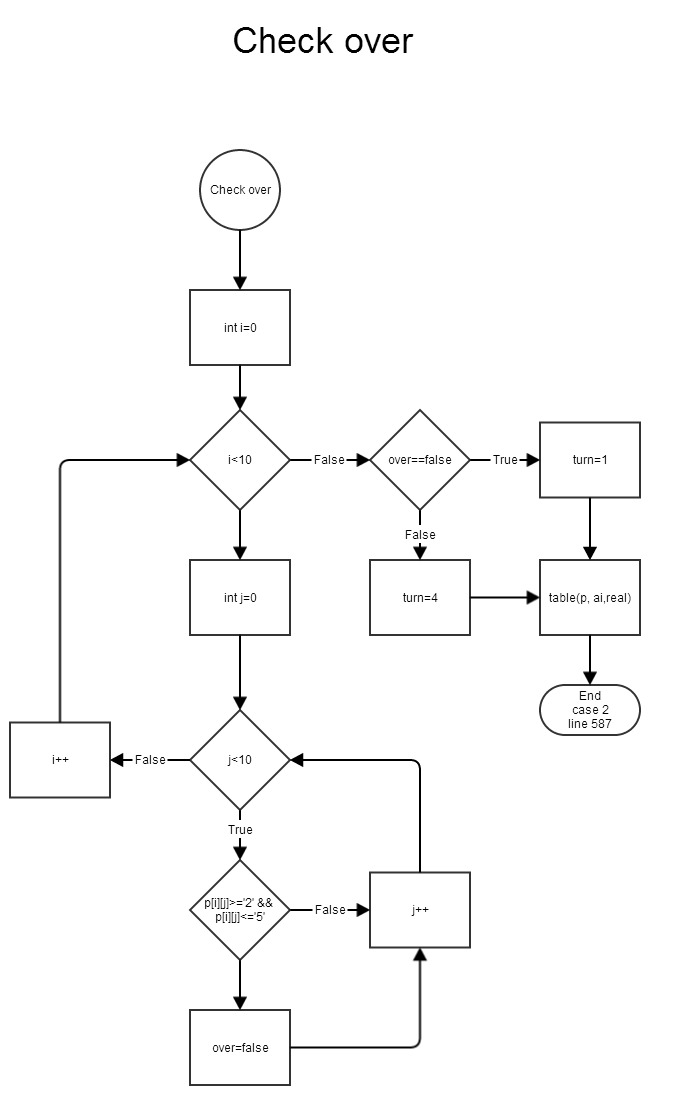
display accuracy rate

ouput sorted array and accuracy to stat.dat

7. Flowchart







8. Code

/\*

\* File: main.cpp

\* Author: Tsz, Kwan

\* Created on July 17, 2014, 5:05 PM

\* Purpose: Battleship for fun

\*/

//System Libraries

#include <cstdlib>

#include <iostream>

#include <ctime>

#include <fstream>

#include <vector>

#include <iomanip>

using namespace std;

//User Libraries

//Global Constant

const int COLS=10;

const int COL=3;

//Function Prototypes

void intro();

void table(char [][COLS],char [][COLS], char [][COLS]);

void pplace(char [][COLS], char [][COLS], char [][COLS], bool &, int [], int &, int &, int &, int &);

void aiplace(char [][COLS], char [][COLS], int &, int &, bool &, int []);

void pause();

int pfire(char [][COLS], char [][COLS], char [][COLS], bool &, bool &, int &, int &, float &, float &);

void getstat(vector<int> &, vector<int> &, vector<char> &, vector<int> &, vector<int> &, vector<char> &, char [][COL], char [][COL], char [][COLS], char [][COLS]);

void sort(char [][COL], char [][COL], vector<char>, vector<char>);

//Execution Begins here

int main(int argc, char\*\* argv) {

//Declare variables

const int ROWS=10, XY=4, ROW=100;

char p[ROWS][COLS]; //player table y,x

char ai[ROWS][COLS]; //ai table show to player

char real[ROWS][COLS]; //real ai table

int unit[5]={5,4,3,2,2}; //unit of ship

int x1,y1,x2,y2; //coordinates to place ship

bool goback=true; //invalid back to random

bool valid; //validation

int count; //use space check validation

int hx=10, hy=10; //first hit coordinate

bool hit; //hit to skip random fire

bool finish=true; //finish one ship back to random

int combo=0, oppcombo=0; //after second hit keep that direction

bool over; //game over

int turn; //switch turn

bool oneend=false; //one side miss/overlap/overside go to opposite direction

bool cross[XY]={true,true,true,true}; //cross 4 boxes around hit

bool crossdone=true; //if true back to random

bool combohit; //keep fire the same direction

float phit=0, pmiss=0; //player hit miss counter

float aihit=0, aimiss=0; //ai hit miss counter

ofstream output;

vector<int> prow,pcol,airow,aicol; //player/ai hit/miss coordinates

vector<char> pr, air; //player/ai hit/miss result

char pvect[ROW][COL]; //player vector array to do sorting

char aivect[ROW][COL]; //ai vector array to do sorting

bool done; //finish fire

int ax,ay; //ai fire coordinate

char cax,cay; //ai fire display in A0 form to player

int hplan; //hit plan after first hit (corss)

cout<<fixed<<showpoint<<setprecision(2);

//reset

srand(static\_cast<unsigned int>(time(0)));

for(int i=0;i<10;i++){ //y coordinates

for(int j=0;j<10;j++){ //x coordinates

p[i][j]=' '; //reset

ai[i][j]=' ';

real[i][j]=' ';

}

}

intro();

//table

table(p, ai, real);

//place ship

pplace(p, ai, real, valid, unit, x1, x2, y1, y2);

//ai part

aiplace(ai, real, x1, y1, valid, unit);

//table

table(p, ai, real);

turn=1;

do{

switch(turn){

//Player fire turn

case 1:{

turn=pfire(p, ai, real, over, valid, x1, y1, phit, pmiss);

break;

}

//ai turn

case 2:{

pause();

over=true;

done=false;

for(int i=0;i<4;i++){

cross[i]=true;

}

do{

done==false;

if(crossdone==true && finish==true && goback==true && done==false && oppcombo==0 && combo==0){

//random fire

// cout<<"random fire\n";

goback=true;

do{

valid=true;

ax=rand()%10;

ay=rand()%10;

if(p[ay][ax]=='O' || p[ay][ax]=='X'){

valid=false;

// cout<<"overlap\n";

}

}while(valid==false);

cax=ax+48;

cay=ay+65;

cout<<"ai fire "<<cay<<cax<<"\n";

if(p[ay][ax]!=' '){

p[ay][ax]='X';

cout<<"Hit!!!\n";

hx=ax;

hy=ay;

hit=true;

crossdone=false;

finish=false;

combo=false;

done=true;

oneend=false;

aihit++;

}

else{

p[ay][ax]='O';

cout<<"Miss...\n";

done=true;

aimiss++;

}

}

//move after hit

if(hit==true && finish==false && crossdone==false && combo==0 && oppcombo==0 && done==false){

do{

// cout<<"random cross\n";

ay=hy;

ax=hx;

//check cross rand

hplan=rand()%4;

if(hplan==0) ay=hy-1;

if(hplan==1) ay=hy+1;

if(hplan==2) ax=hx-1;

if(hplan==3) ax=hx+1;

// cout<<"hplan = "<<hplan<<endl;

//check over size

if(ay<0 || ay>9 || ax<0 || ax>9){

// cout<<"Out table\n";

cross[hplan]=false;

}

else if(p[ay][ax]=='X' || p[ay][ax]=='O'){

// cout<<"overlap\n";

cross[hplan]=false;

}

if(cross[0]==cross[1] && cross[1]==cross[2] && cross[2]==cross[3] && cross[0]==false){

// cout<<"test all 4 but invalid\n";

crossdone=true;

finish=true;

goback=true;

}

}while(crossdone==false && cross[hplan]==false);

//valid

if(crossdone==false){

// cout<<"check hit or miss by cross rand xy\n";

cax=ax+48;

cay=ay+65;

cout<<"ai fire "<<cay<<cax<<"\n";

if(p[ay][ax]!=' '){

p[ay][ax]='X';

cout<<"Hit!!!\n";

done=true;

combo++;

crossdone=true;

aihit++;

}

else{

p[ay][ax]='O';

cout<<"Miss...\n";

done=true;

aimiss++;

}

}

else{

// cout<<"crossdone=true, go back to rand \n";

goback=true;

}

}

else if(combo>0 && oneend==false && done==false && crossdone==true){ //continue check

// cout<<"second hit\n";

valid=true;

if(hx==ax){

// cout<<"same x\n";

if(hy>ay) ay=hy-combo-1;

else ay=hy+combo+1;

if(ay<0 || ay >9){

valid=false;

}

if(valid==true){

if(p[ay][ax]=='X' || p[ay][ax]=='O'){

valid=false;

}

if(p[ay][ax]=='O'){

finish=true;

goback=true;

crossdone=true;

combo=0;

}

if(valid==true){

cax=ax+48;

cay=ay+65;

cout<<"ai fire "<<cay<<cax<<"\n";

if(p[ay][ax]!=' '){

p[ay][ax]='X';

cout<<"Hit!!!\n";

done=true;

combo++;

aihit++;

}

else{

p[ay][ax]='O';

cout<<"Miss...\n";

done=true;

oneend=true;

oppcombo++;

aimiss++;

}

}

}

else{ //check ->GO TO OPPCOMBO

// cout<<"next xy invalid change to opposite side\n";

combo=0;

oneend=true;

crossdone=true;

oppcombo++;

combohit=false;

}

}

if(hy==ay){

// cout<<"same y\n";

if(hx>ax) ax=hx-combo-1;

else ax=hx+combo+1;

if(ax<0 || ax >9){

valid=false;

combo=0;

goback=true;

finish=true;

}

if(valid==true){

if(p[ay][ax]=='X' || p[ay][ax]=='O'){

valid=false;

finish=true;

goback=true;

}

if(valid==true){

cax=ax+48;

cay=ay+65;

cout<<"ai fire "<<cay<<cax<<"\n";

if(p[ay][ax]!=' '){

p[ay][ax]='X';

combo++;

done=true;

aihit++;

}

else{

p[ay][ax]='O';

cout<<"Miss...\n";

done=true;

oneend=true;

oppcombo++;

combo=0;

combohit=false;

// cout<<"oneend==true\n";

// cout<<"done==true\n";

aimiss++;

}

}

}

if(valid==false){ //GO TO OPPCOMBO

// cout<<"next xy inlvalid change to other side\n";

combo=0;

oneend=true;

crossdone=true;

oppcombo++;

combohit=false;

}

}

}

else if(oppcombo>0 && oneend==true && done==false){ //check other side

// cout<<"one side end check other side\n";

// cout<<"oppcombo = "<<oppcombo<<endl;

if(hx==ax){

// cout<<"same X\n";

if(combohit==false){

if(hy>ay) ay=hy+oppcombo;

else ay=hy-oppcombo;

}

else{

if(ay>hy) ay=hy+oppcombo;

else ay=hy-oppcombo;

}

// cout<<ay<<ax<<endl;

if(ay<0 || ay>9 || p[ay][ax]=='O' || p[ay][ax]=='X'){

oppcombo=0;

goback=true;

finish=true;

crossdone=true;

combo=0;

done=false;

// cout<<"overlap or oversize\n";

}

else{

cax=ax+48;

cay=ay+65;

cout<<"ai fire "<<cay<<cax<<"\n";

if(p[ay][ax]!=' '){

p[ay][ax]='X';

cout<<"Hit!!!\n";

done=true;

oppcombo+=1;

combohit=true;

aihit++;

}

else{

p[ay][ax]='O';

cout<<"Miss...\n";

done=true;

combo=0;

oppcombo=0;

finish=true;

goback=true;

crossdone=true;

aimiss++;

}

}

}

else if(hy==ay){

// cout<<"same y\n";

if(combohit==false){

if(hx>ax) ax=hx+oppcombo;

else ax=hx-oppcombo;

}

else{

if(ax>hx) ax=hx+oppcombo;

else ax=hx-oppcombo;

}

// cout<<ay<<ax<<endl;

if(ax<0 || ax>9 || p[ay][ax]=='O' || p[ay][ax]=='X'){

oppcombo=0;

goback=true;

finish=true;

crossdone=true;

combo=0;

done=false;

// cout<<"overlap or oversize\n";

}

else{

cax=ax+48;

cay=ay+65;

cout<<"ai fire "<<cay<<cax<<"\n";

if(p[ay][ax]!=' '){

p[ay][ax]='X';

cout<<"Hit!!!\n";

done=true;

oppcombo+=1;

combohit=true;

aihit++;

}

else{

p[ay][ax]='O';

cout<<"Miss...\n";

done=true;

combo=0;

oppcombo=0;

finish=true;

goback=true;

crossdone=true;

aimiss++;

}

}

}

}

}while(done==false);

for(int i=0;i<10;i++){ //check over

for(int j=0;j<10;j++){

if(p[i][j]>='2' && p[i][j]<='5')

over=false;

}

}

if(over==false) turn=1;

else turn=4;

//table

table(p, ai, real);

break;

}

case 3:{

cout<<"You win \n";

turn=5;

break;

}

case 4:{

cout<<"You lose \n";

turn=5;

break;

}

}

}while(turn<5);

getstat(prow, pcol, pr, airow, aicol, air, pvect, aivect, p, ai);

sort( pvect, aivect, pr, air);

cout<<"Your accuracy is ";

cout<<100\*phit/(phit+pmiss);

cout<<"%\n";

cout<<"AI accuracy is ";

cout<<100\*aihit/(aihit+aimiss);

cout<<"%\n\n";

cout<<endl;

output.open("stat.dat");

if(output.fail()){

cout<<"Output file opening failed.\n";

}

output<<"Your accuracy is ";

output<<100\*phit/(phit+pmiss);

output<<"%\n";

output<<"AI accuracy is ";

output<<100\*aihit/(aihit+aimiss);

output<<"%\n\n";

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

output<<pvect[i][0]<<pvect[i][1];

output<<" ";

output<<pvect[i][2];

output<<" ";

output<<aivect[i][0]<<aivect[i][1];

output<<" ";

output<<aivect[i][2];

output<<endl;

}

output.close();

return 0;

}

void intro(){

cout<<"Battleship!\n";

cout<<"You have 5 ship to place\n";

cout<<"5 units ship\*1 55555, 4 units ship\*1 4444\n";

cout<<"3 units ship\*1 333, 2 units ships\*2 22, 22\n";

}

void table(char p[][COLS], char ai[][COLS], char real[][COLS]){

//table

cout<<" PLAYER 1 A.I.\n";

cout<<" 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9\n";

cout<<" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

for(int i=0;i<10;i++){

char row=i+65;

cout<<row<<"| ";

for(int j=0;j<10;j++){

cout<<p[i][j]<<" | ";

}

cout<<" "<<row<<"| ";

for(int j=0;j<10;j++){

cout<<ai[i][j]<<" | ";

}

cout<<endl;

cout<<" ---------------------------------------- ----------------------------------------\n";

}

}

void pause(){

time\_t start, end;

start=time(0);

do{

end=time(0);

}while(difftime(end,start)<1);

}

void pplace(char p[][COLS], char ai[][COLS], char real[][COLS], bool &valid, int unit[], int &x1, int &x2, int &y1, int &y2){

int count;

int max, min; //replace the coordinates to place ship

string place; //x,y to place ship start and end coordinates

//place ship

for(int q=0;q<5;q++){

do{

do{

do{

count=0;

valid=true; //reset

cout<<"Choose the coordinates to place the ";

cout<<unit[q]<<"-unit ship with A1A5 form : ";

cin>>place;

if(place.size()!=4){ //check size

cout<<"size\n";

valid=false;

}

if(place[0]<'A' || place[0]>'J' || place[2]<'A' || place[2]>'J'){

valid=false;

}

if(valid==false){

cout<<"Invalid input\n";

}

}while(valid==false);

cout<<place[0]-65<<place[1]-48<<place[2]-65<<place[3]-48<<endl;

y1=place[0]-65;

y2=place[2]-65;

x1=place[1]-48;

x2=place[3]-48;

cout<<y1<<x1<<y2<<x2<<endl;

if(y1==y2){ //x is same

if(abs(x1-x2)!=unit[q]-1){ //check unit invalid

cout<<"x unit\n";

valid=false;

}

else{ //valid

if(x1>x2){ //check which larger

max=x1;

min=x2;

}

else{

max=x2;

min=x1;

}

cout<<"max="<<max<<endl;

cout<<"min="<<min<<endl;

cout<<"p"<<y1<<endl;

for(int k=min;k<=max;k++){ //check overlap

if(p[y1][k]==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

cout<<"overlap\n";

}

if(valid==true){

for(int k=min;k<=max;k++){

p[y1][k]=unit[q]+48;

}

}

}

}

if(x1==x2){ //y is same

if(abs(y1-y2)!=unit[q]-1){ //check unit

cout<<"y unit\n";

valid=false;

}

else{ //valid

if(y1>y2){

max=y1;

min=y2;

}

else{

max=y2;

min=y1;

}

cout<<"max="<<max<<endl;

cout<<"min="<<min<<endl;

cout<<"p"<<y1<<endl;

for(int k=min;k<=max;k++){

if(p[k][x1]==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

cout<<"overlap\n";

}

if(valid==true){

for(int k=min;k<=max;k++){

p[k][x1]=unit[q]+48;

}

}

}

}

if(x1!=x2 && y1!=y2){

valid=false;

cout<<"horizontal/vertical\n";

}

}while(valid==false);

cout<<count<<endl;

}while(valid==false);

//table

table(p, ai, real);

}

}

void aiplace(char ai[][COLS], char real[][COLS], int &x1, int &y1, bool &valid, int unit[]){

int count;

int pos; //place horizontal/vertical(ai)

for(int q=0;q<5;q++){

do{

valid=true;

count=0;

//random coordinates

y1=rand()%(10-unit[q]); //won't over size

x1=rand()%(10-unit[q]);

pos=rand()%2;

if(pos==0){ //0 horizontal

for(int k=y1;k<y1+unit[q];k++){

if(real[k][x1]==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

}

if(valid==true){

for(int k=y1;k<y1+unit[q];k++){

real[k][x1]=unit[q]+48;

}

}

}

else{ //1 vertical

for(int k=x1;k<x1+unit[q];k++){

if(real[y1][k]==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

}

if(valid==true){

for(int k=x1;k<x1+unit[q];k++){

real[y1][k]=unit[q]+48;

}

}

}

}while(valid==false);

}

}

int pfire(char p[][COLS], char ai[][COLS], char real[][COLS], bool &over, bool &valid, int &x1, int &y1, float &phit, float &pmiss){

string fire; //player fire;

over=true;

do{

valid=true;

cout<<"Your turn, please enter a coordinate to fire in A0 form :";

cin>>fire;

if(fire.length()!=2){

valid=false;

cout<<"size\n";

}

if(fire[0]<'A' || fire[0]>'J' || fire[1]<'0' || fire[1]>'9'){

valid=false;

cout<<"x/y\n";

}

y1=fire[0]-65;

x1=fire[1]-48;

if(real[y1][x1]=='O' || real[y1][x1]=='X'){

valid=false;

cout<<"overlap\n";

}

}while(valid==false);

//hit

if(real[y1][x1]>='2' && real[y1][x1]<='5'){

cout<<"Hit!!!\n";

real[y1][x1]='X';

ai[y1][x1]='X';

phit++;

}

else{

cout<<"Miss....\n";

real[y1][x1]='O';

ai[y1][x1]='O';

pmiss++;

}

//table

table(p, ai, real);

for(int i=0;i<10;i++){

for(int j=0;j<10;j++){

if(real[i][j]>='2' && real[i][j]<='5')

over=false;

}

}

if(over==true){

return 3;

}

else return 2;

}

void getstat(vector<int> &prow, vector<int> &pcol, vector<char> &pr, vector<int> &airow, vector<int> &aicol, vector<char> &air, char pvect[][COL], char aivect[][COL], char p[][COLS], char ai[][COLS]){

for(int i=0;i<10;i++){

for(int j=0;j<10;j++){

if(p[i][j]=='X'){

prow.push\_back(i);

pcol.push\_back(j);

pr.push\_back(p[i][j]);

}

if(ai[i][j]=='X'){

airow.push\_back(i);

aicol.push\_back(j);

air.push\_back(ai[i][j]);

}

}

for(int j=0;j<10;j++){

if(p[i][j]=='O'){

prow.push\_back(i);

pcol.push\_back(j);

pr.push\_back(p[i][j]);

}

if(ai[i][j]=='O'){

airow.push\_back(i);

aicol.push\_back(j);

air.push\_back(ai[i][j]);

}

}

}

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

pvect[i][0]=prow[i]+65;

pvect[i][1]=pcol[i]+48;

pvect[i][2]=pr[i];

aivect[i][0]=airow[i]+65;

aivect[i][1]=aicol[i]+48;

aivect[i][2]=air[i];

}

}

void sort(char pvect[][COL], char aivect[][COL], vector<char> pr, vector<char> air){

bool swap; //sorting swap

char temp;

do{

swap=false;

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

if(pvect[i][2]<pvect[i+1][2]){

for(int k=0;k<3;k++){

temp=pvect[i][k];

pvect[i][k]=pvect[i+1][k];

pvect[i+1][k]=temp;

swap=true;

}

}

}

}while(swap);

do{

swap=false;

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

if(pvect[i][0]>pvect[i+1][0] && pvect[i][2]==pvect[i+1][2]){

for(int k=0;k<3;k++){

temp=pvect[i][k];

pvect[i][k]=pvect[i+1][k];

pvect[i+1][k]=temp;

swap=true;

}

}

}

}while(swap);

do{

swap=false;

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

if(pvect[i][1]>pvect[i+1][1] && pvect[i][2]==pvect[i+1][2] && pvect[i][1]==pvect[i+1][1]){

for(int k=0;k<3;k++){

temp=pvect[i][k];

pvect[i][k]=pvect[i+1][k];

pvect[i+1][k]=temp;

swap=true;

}

}

}

}while(swap);

do{

swap=false;

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

if(aivect[i][2]<aivect[i+1][2]){

for(int k=0;k<3;k++){

temp=aivect[i][k];

aivect[i][k]=aivect[i+1][k];

aivect[i+1][k]=temp;

swap=true;

}

}

}

}while(swap);

do{

swap=false;

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

if(aivect[i][0]>aivect[i+1][0] && aivect[i][2]==aivect[i+1][2]){

for(int k=0;k<3;k++){

temp=aivect[i][k];

aivect[i][k]=aivect[i+1][k];

aivect[i+1][k]=temp;

swap=true;

}

}

}

}while(swap);

do{

swap=false;

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

if(aivect[i][1]>aivect[i+1][1] && aivect[i][2]==aivect[i+1][2] && aivect[i][1]==aivect[i+1][1]){

for(int k=0;k<3;k++){

temp=aivect[i][k];

aivect[i][k]=aivect[i+1][k];

aivect[i+1][k]=temp;

swap=true;

}

}

}

}while(swap);

}