Battleship

Project2

CSC- 17A – 48130

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)…………………….……...7

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

6. Pseudo Code……………………………………….7-10

7. Flowchart…………………………………………11-15

8. Code……………………………………………...16-46

1. Introduction

**Rules and Gameplay**

The players can choose the size of the table which 8x8 to 10x10. After the player choose the size, two tables created . 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 acts like a human player. I want the AI check the coordinates around the hit coordinate, and keep fire when it 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 it records and shows “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 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. It is possible the tables which are larger than 10x10, but there are only 0-9 for digits, and it is not a good idea to use low case characters with digits because it is confusing. 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()%num-5(num is the size of the table). 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 | num | size of 2d dynamic array | main | 40 |
|  | turn=1 | menu turns | main | 41 |
|  | n | function getN return var | main | 103 |
|  | count |  | main | 212 |
|  | max, min |  | main | 213 |
|  | turn | in function aifire return var | main | 429 |
|  | hplan | hit plan after first hit (cross) | main | 431 |
|  | unit[5] | units of the ships in an array | Player | 15 |
|  | x1, x2, y1, y2 | 2 coordinates to place ship | Player | 16 |
|  | hx, hy, x, y | current, pre fire coordinates | AI | 19 |
|  | oppcombo | opposite side combo | AI | 21 |
|  | combo |  | AI | 22 |
| float | pttl=0, aittl=0 | both total number of fire | main | 743 |
|  | pac, aiac | accuracy | main | 744 |
|  | hit | player hit counter | Player | 20 |
|  | miss | player miss counter | Player | 21 |
|  | hit, miss | ai hit miss counter | AI | 29 |
| char | temp |  | main | 115 |
|  | row |  | main | 187 |
|  | \*\*pt | player table | Player | 12 |
|  | \*\*fake, \*\*real | ai fake real tales | AI | 11 |
|  | cx, cy | ai coordinates in char | AI | 23 |
| string | temp |  | main | 95 |
|  | place | player place input type | main | 214 |
|  | fire | player fire input type | main | 372 |
| bool | invalid=false | num validation | main | 104 |
|  | valid | string validation | main | 215 |
|  | digit | isdidit validation | main | 216 |
|  | valid | ai place validation | main | 329 |
|  | over=true | game over Boolean | main | 373 |
|  | valid | player fire validation | main | 374 |
|  | valid | ai fire validation | main | 430 |
|  | over | ai over boolean | AI | 14 |
|  | done | finish fire | AI | 15 |
|  | cross[4] | cross 4 boxes around hit | AI | 16 |
|  | crossdone | if true back to random | AI | 17 |
|  | goback | invalid back to random | AI | 18 |
|  | finish | combo finish Boolean | AI | 20 |
|  | hit | hit Boolean | AI | 24 |
|  | oneend | one side finish | AI | 25 |
|  | combohit | keep fire the same direction | AI | 26 |
| fstream | io | i/o file in function intro | main | 93 |
|  | io | i/o file in function init | main | 114 |
|  | io | i/o file in function gestates | main | 742 |
| time\_t | start, end | delay display ai fire | main | 760 |
| PlayerT | pT | Player table | main | 42 |
| PlayerG | pG | Player game data | main | 43 |
| PlayerS | \*ps | Player States | main | 44 |
| AIT | ait | AI table | main | 45 |
| AIG | ai | AI game data | main | 46 |
| AIS | \*as | AI game states | main | 47 |

4. Topic Covered (Checklist)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chapter | type | code | cpp | line |
| Memory Allocation | char\*\* | pT.pt = new char\*[num]; | main | 147 |
|  |  | for(int i=0;i<num;i++){ | main | 150 |
|  |  | pT.pt[i] = new char[num]; | main | 151 |
|  |  | } | main | 154 |
| delete 2d dynamic arr |  | for (int i=0;i<num;i++){ | main | 80 |
|  |  | delete[] pT.pt[i]; | main | 81 |
|  |  | } | main | 84 |
|  |  | delete[] pT.pt; | main | 85 |
| function with struct |  | void init(PlayerT &, AIT &, int, PlayerG &, PlayerS &, AIG &, AIS &); | main | 30 |
|  |  | void table(char \*\*, char \*\*, char \*\*, int); | main | 31 |
| pointer notation |  | if(\*(\*(pt+pG.y1)+k)==' '){ | main | 267 |
| structure pointer | PlayerS | PlayerS \*ps | main | 44 |
| delete struct pointer |  | delete ps, as; | main | 88 |
| cctype | isdigit | if(isdigit(place[1]) && isdigit(place[3])){ | main | 234 |
| input binary | input | io.open("unit.txt", ios::in | ios::binary); | main | 116 |
|  | ouput | io.open("rank.txt", ios::out | ios::binary); | main | 749 |
| struct |  | struct PlayerT | Player | 14 |
|  | array | int unit[5]; | Player | 15 |
| difftime |  | }while(difftime(end,start)<1); | main | 772 |

5. Libraries included

* <cstdlib>
* <iostream>
* <ctime>
* <fstream>
* <iomanip>
* <cctype>
* Player.h
* AI.h

6. Pseudo Code

Ask the size of the table

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 fire)**

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, turn=5

**case 4**

Player lose, turn=5

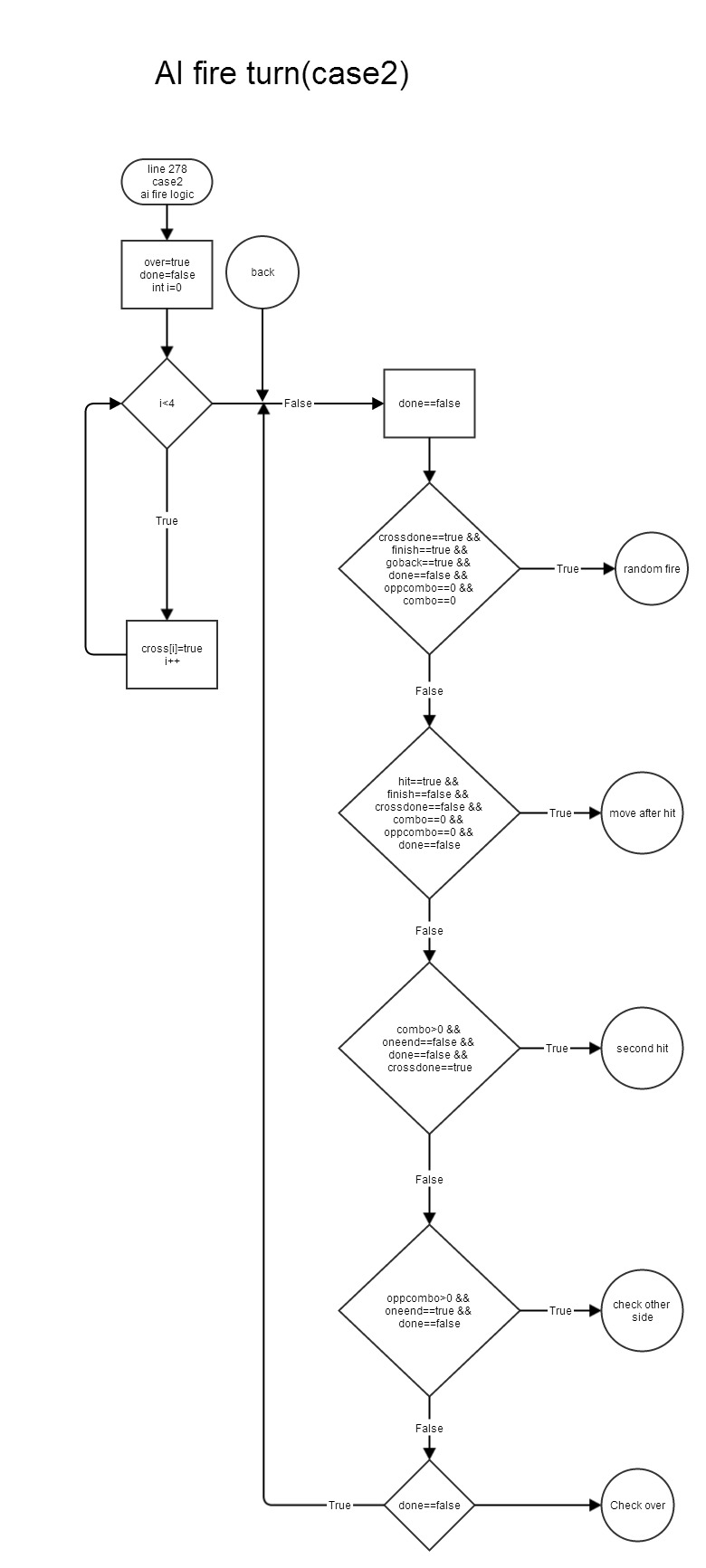
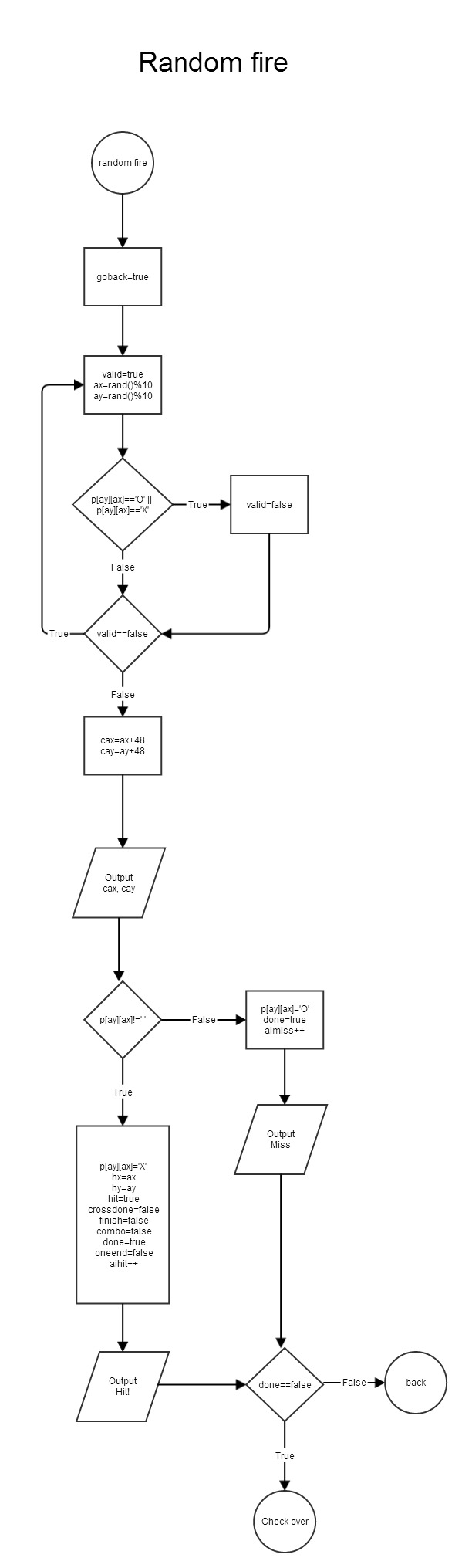
if(turn<5) keep looping the case

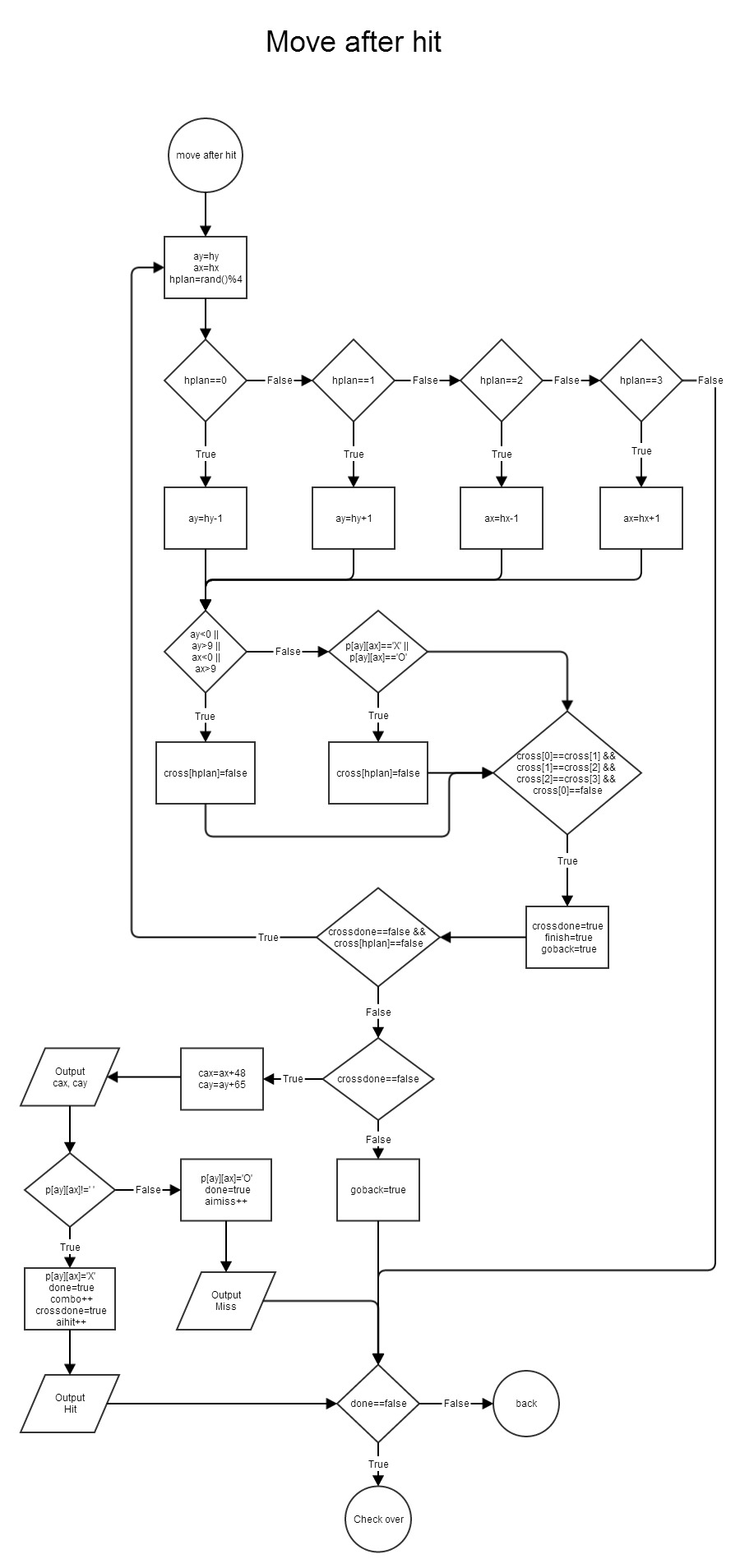
scan player and ai table

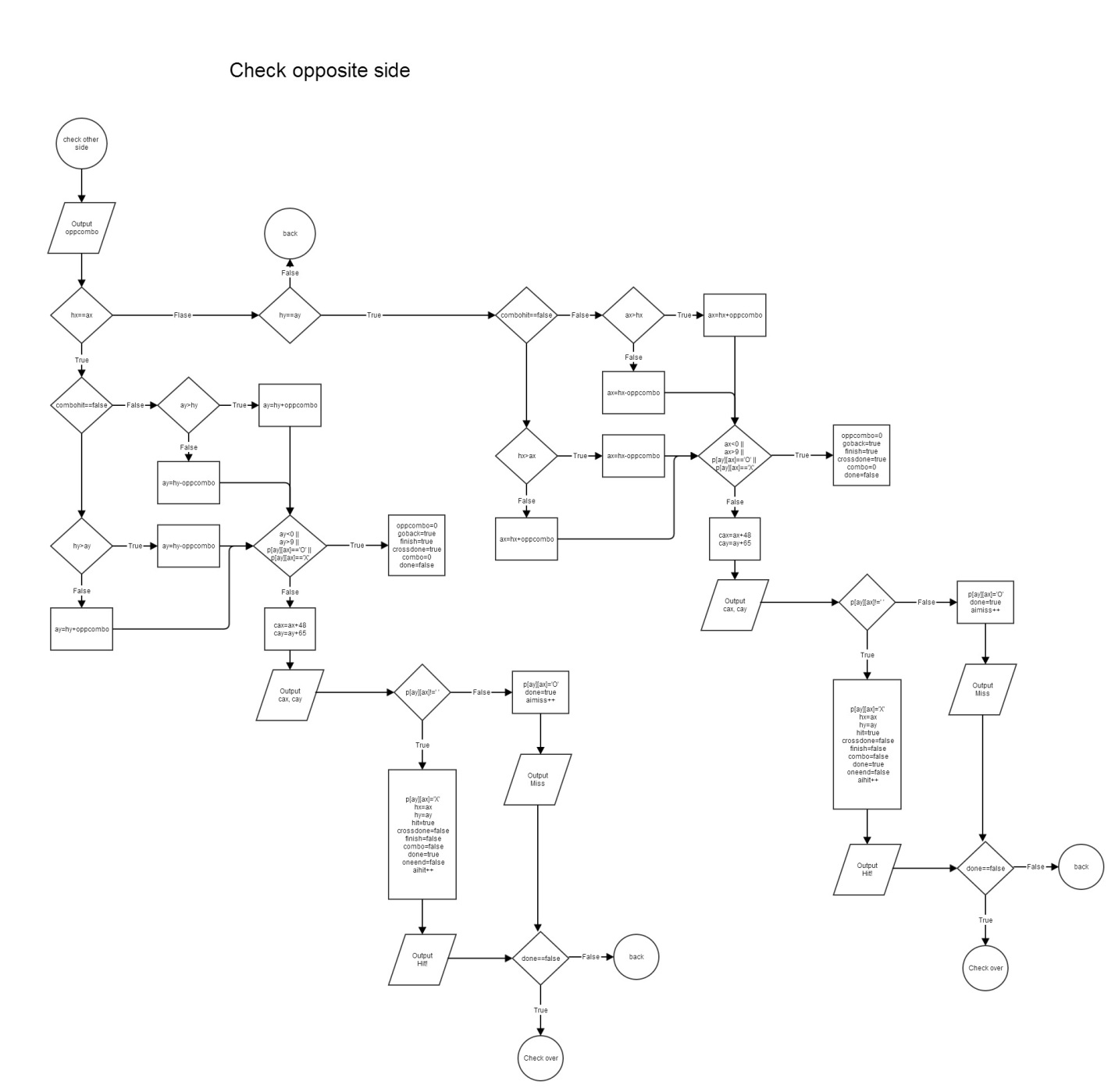
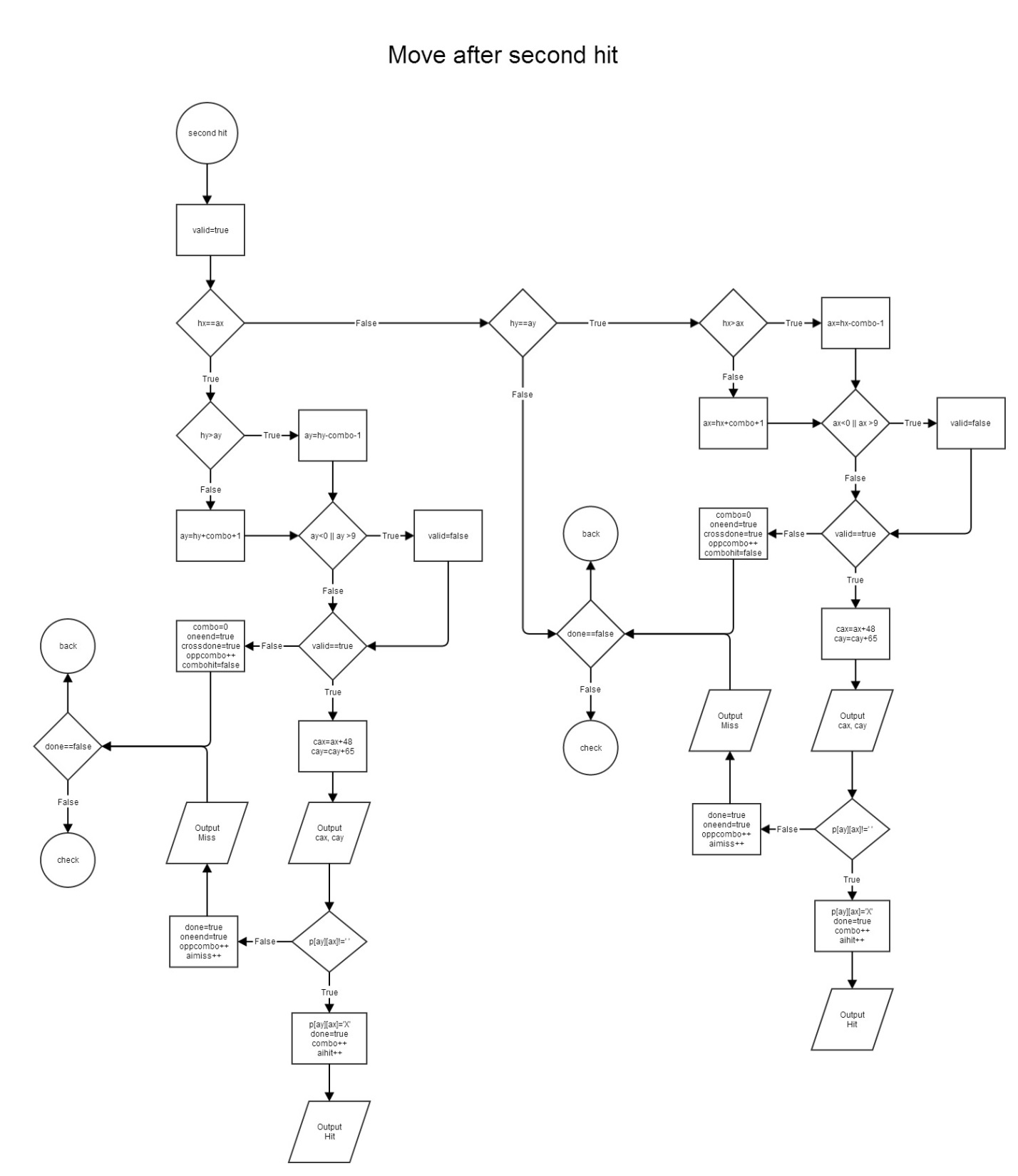
calculate accuracy hit/(hit+miss)

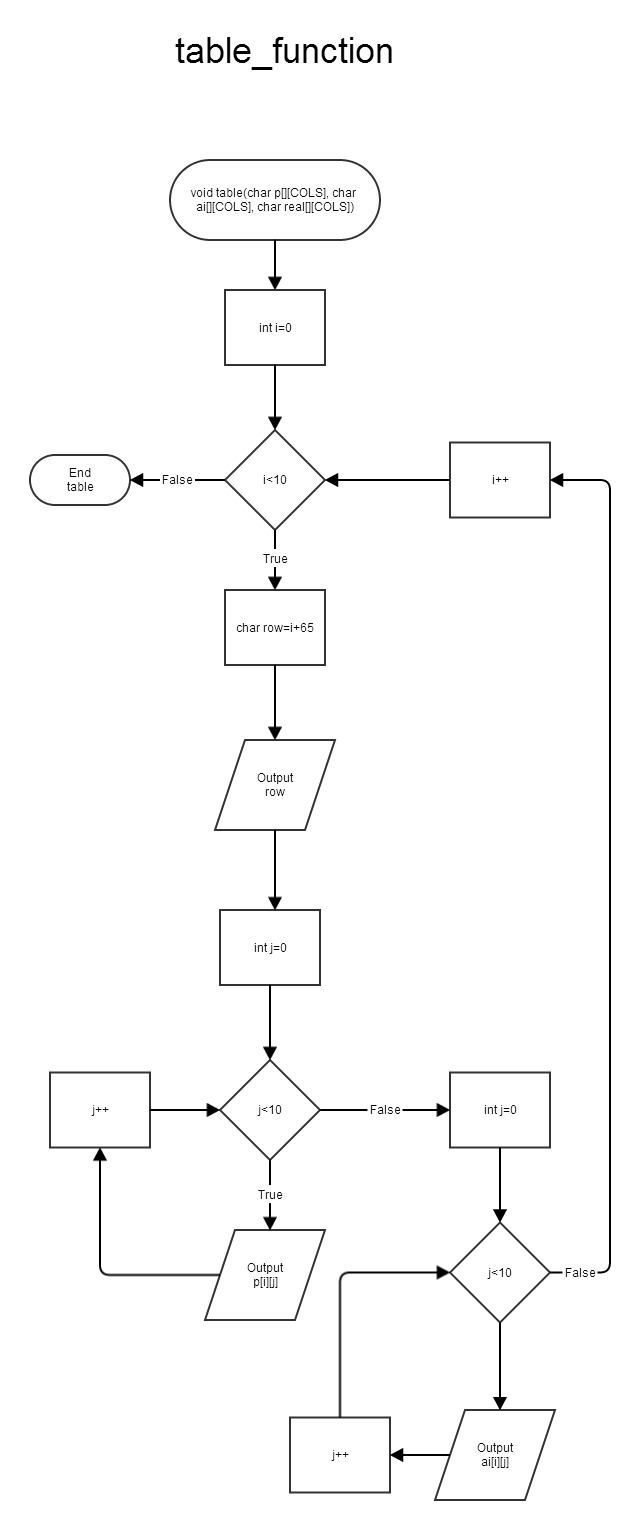
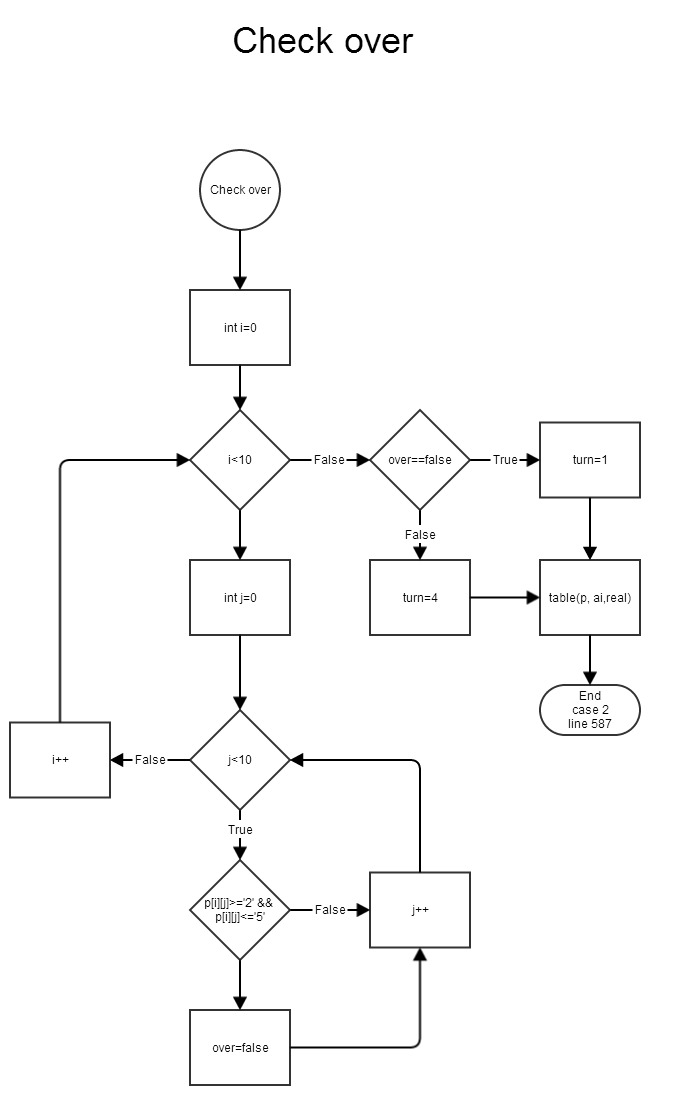
display player rank

output to rank.txt

7. Flowchart







8. Code - main

/\*

\* File: main.cpp

\* Author: Tsz, Kwan

\* Created on September 30, 2014, 10:33 AM

\* Purpose: CSC17A\_Porj1\_Battleship

\* 2D\_Dynamic\_array

\* Structure

\* cctype\_binary

\*/

//System libraries

#include <cstdlib>

#include <iostream>

#include <iomanip>

#include <ctime>

#include <fstream>

#include <cctype>

using namespace std;

//User Libraries

#include "Player.h"

#include "AI.h"

//Global Constant

//Function Prototypes

void intro();

int getN();

void pause();

void init(PlayerT &, AIT &, int, PlayerG &, PlayerS \*, AIG &, AIS \*);

void table(char \*\*, char \*\*, char \*\*, int);

void pplace(char \*\*, char \*\*, char \*\*, PlayerG &, int);

void aiplace(char \*\*, char \*\*, char \*\*, int [], int, int, int);

int pfire(char \*\*, char \*\*, char \*\*, PlayerG &, PlayerS \*, int);

int aifire(char \*\*, char \*\*, char \*\*, AIG &, AIS \*, int);

void getstates(PlayerS \*);

//System Begins Here

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

intro();

int num=getN();

int turn=1;

PlayerT pT;

PlayerG pG;

PlayerS \*ps = new PlayerS;

AIT ait;

AIG ai;

AIS \*as = new AIS;

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

//reset

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

init(pT, ait, num, pG, ps, ai, as);

table(pT.pt, ait.fake, ait.real, num);

//player prepare

pplace(pT.pt, ait.fake, ait.real, pG, num);

aiplace(pT.pt, ait.fake, ait.real, pG.unit, pG.x1, pG.y1, num);

//game start

do{

switch(turn){

case 1:{

turn=pfire(pT.pt, ait.fake, ait.real, pG, ps, num);

break;

}

case 2:{

turn=aifire(pT.pt, ait.fake, ait.real, ai, as, num);

break;

}

case 3:{

cout<<"You win!\n\n";turn=5;

break;

}

case 4:{

cout<<"You lose!\n\n";turn=5;

break;

}

}

}while(turn<5);

getstates(ps);

//free memory

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

delete[] pT.pt[i];

delete[] ait.real[i];

delete[] ait.fake[i];

}

delete[] pT.pt;

delete[] ait.real;

delete[] ait.fake;

delete ps, as;

return 0;

}

void intro(){

fstream io;

io.open("intro.txt", ios::in);

if(io.is\_open()){

string temp;

while(getline(io, temp))

cout<<temp<<endl;

io.close();

}

}

int getN(){

int n;

bool invalid=false;

do{

cout<<"Please choose the size of the table (8-10) :";

cin>>n;

if(n<8 || n>10)

invalid=true;

}while(invalid);

return n;

}

void init(PlayerT &pT,AIT &ait, int num, PlayerG &pG, PlayerS \*ps, AIG &ai, AIS \*as){

fstream io;

char temp;

io.open("unit.txt", ios::in | ios::binary);

if(io.is\_open()){

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

io.read(&temp, sizeof(temp));

pG.unit[i]=temp-48;

}

io.close();

}

//init by default

else{

cout<<"Default\n";

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

pG.unit[i]=5-i;

pG.unit[4]=2;

}

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

ai.cross[i]=true;

ai.crossdone=true;

ai.goback=true;

ai.hx=10;

ai.hy=10;

ai.finish=true;

ai.goback=true;

ai.oppcombo=0;

ai.combo=0;

ai.oneend=false;

as->hit=0;

as->miss=0;

ps->hit=0;

ps->miss=0;

//init table

pT.pt = new char\*[num];

ait.fake = new char\*[num];

ait.real = new char\*[num];

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

pT.pt[i] = new char[num];

ait.fake[i] = new char[num];

ait.real[i] = new char[num];

}

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

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

pT.pt[i][j]=' ';

ait.fake[i][j]=' ';

ait.real[i][j]=' ';

}

}

}

void table(char \*\*pt, char \*\*fake, char \*\*real, int num){

//table

//space 38, 3

cout<<" PLAYER 1"<<setw(num\*4+4)<<"A.I.\n";

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

cout<<setw(4)<<i;

}

cout<<" ";

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

cout<<setw(4)<<i;

}

cout<<endl;

//third line

cout<<" ";

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

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

}

cout<<" ";

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

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

}

cout<<endl;

//forth to num line

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

char row=i+65;

cout<<row<<"| ";

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

cout<<pt[i][j];

cout<<" "<<"| ";

}

cout<<" "<<"| ";

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

cout<<fake[i][k];

cout<<" "<<"| ";

}

cout<<endl;

cout<<" ";

for(int l=0;l<num;l++){

cout<<"----";

}

cout<<" ";

for(int m=0;m<num;m++){

cout<<"----";

}

cout<<endl;

}

}

void pplace(char \*\*pt, char \*\*fake, char \*\*real, PlayerG &pG, int num){

int count;

int max, min;

string place;

bool valid;

bool digit;

//place ship

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

do{

do{

do{

count=0;

digit=false;

valid=true; //reset

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

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

cin>>place;

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

cout<<"size\n";

valid=false;

}

digit=false;

if(isdigit(place[1]) && isdigit(place[3])){

digit=true;

}

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

valid=false;

}

if(valid==false || digit==false){

cout<<"Invalid input\n";

}

}while(valid==false || digit==false);

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

pG.y1=place[0]-65;

pG.y2=place[2]-65;

pG.x1=place[1]-48;

pG.x2=place[3]-48;

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

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

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

cout<<"x unit\n";

valid=false;

}

else{ //valid

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

max=pG.x1;

min=pG.x2;

}

else{

max=pG.x2;

min=pG.x1;

}

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

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

cout<<"p"<<pG.y1<<endl;

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

if(\*(\*(pt+pG.y1)+k)==' '){

count++;

}

}

if(count!=pG.unit[q]){

valid=false;

cout<<"overlap\n";

}

if(valid==true){

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

\*(\*(pt+pG.y1)+k)=pG.unit[q]+48;

}

}

}

}

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

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

cout<<"y unit\n";

valid=false;

}

else{ //valid

if(pG.y1>pG.y2){

max=pG.y1;

min=pG.y2;

}

else{

max=pG.y2;

min=pG.y1;

}

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

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

cout<<"p"<<pG.y1<<endl;

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

if(\*(\*(pt+k)+pG.x1)==' '){

count++;

}

}

if(count!=pG.unit[q]){

valid=false;

cout<<"overlap\n";

}

if(valid==true){

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

\*(\*(pt+k)+pG.x1)=pG.unit[q]+48;

}

}

}

}

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

valid=false;

cout<<"horizontal/vertical\n";

}

}while(valid==false);

cout<<count<<endl;

}while(valid==false);

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

//table

table(pt, fake, real, num);

}

}

void aiplace(char \*\*pt, char \*\*fake, char \*\*real, int unit[], int x1,int y1, int num){

int count, pos;

bool valid;

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

do{

valid=true;

count=0;

//random coordinates

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

x1=rand()%(num-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 \*\*pt, char \*\*fake, char \*\*real, PlayerG &pG, PlayerS \*ps, int num){

string fire; //player fire;

bool over=true;

bool valid;

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";

}

//fire[1]<'0' || fire[1]>'9' ||

if(fire[0]<'A' || fire[0]>'J' || fire[1]<'0' || fire[1]>num+48-1){

valid=false;

cout<<"x/y\n";

}

pG.y1=fire[0]-65;

pG.x1=fire[1]-48;

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

valid=false;

cout<<"overlap\n";

}

}while(valid==false);

//hit

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

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

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

fake[pG.y1][pG.x1]='X';

ps->hit++;

}

else{

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

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

fake[pG.y1][pG.x1]='O';

ps->miss++;

}

//table

table(pt, fake, real, num);

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

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

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

over=false;

}

}

if(over==true){

// cout<<"3\n";

return 3;

}

else{

// cout<<"2\n";

return 2;

}

}

int aifire(char \*\*pt, char \*\*fake, char \*\*real, AIG &ai, AIS \*as, int num){

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

cout<<"aifire\n";

pause();

int turn;

bool valid;

int hplan;

ai.over=true;

ai.done=false;

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

ai.cross[i]=true;

}

do{

ai.done=false;

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

//random fire

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

ai.goback=true;

do{

valid=true;

ai.x=rand()%num;

ai.y=rand()%num;

if(pt[ai.y][ai.x]=='O' || pt[ai.y][ai.x]=='X'){

valid=false;

cout<<"overlap\n";

}

}while(valid==false);

ai.cx=ai.x+48;

ai.cy=ai.y+65;

cout<<"ai fire "<<ai.cy<<ai.cx<<"\n";

if(pt[ai.y][ai.x]!=' '){

pt[ai.y][ai.x]='X';

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

ai.hx=ai.x;

ai.hy=ai.y;

ai.hit=true;

ai.crossdone=false;

ai.finish=false;

ai.combo=false;

ai.done=true;

ai.oneend=false;

as->hit++;

}

else{

pt[ai.y][ai.x]='O';

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

ai.done=true;

as->miss++;

}

}

//move after hit

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

do{

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

ai.y=ai.hy;

ai.x=ai.hx;

//check cross rand

hplan=rand()%4;

if(hplan==0) ai.y=ai.hy-1;

if(hplan==1) ai.y=ai.hy+1;

if(hplan==2) ai.x=ai.hx-1;

if(hplan==3) ai.x=ai.hx+1;

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

//check over size

if(ai.y<0 || ai.y>num-1 || ai.x<0 || ai.x>num-1){

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

ai.cross[hplan]=false;

}

else if(pt[ai.y][ai.x]=='X' || pt[ai.y][ai.x]=='O'){

// cout<<"overlap\n";

ai.cross[hplan]=false;

}

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

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

ai.crossdone=true;

ai.finish=true;

ai.goback=true;

}

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

//valid

if(ai.crossdone==false){

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

ai.cx=ai.x+48;

ai.cy=ai.y+65;

cout<<"ai fire "<<ai.cy<<ai.cx<<"\n";

if(pt[ai.y][ai.x]!=' '){

pt[ai.y][ai.x]='X';

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

ai.done=true;

ai.combo++;

ai.crossdone=true;

as->hit++;

}

else{

pt[ai.y][ai.x]='O';

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

ai.done=true;

as->miss++;

}

}

else{

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

ai.goback=true;

}

}

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

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

valid=true;

if(ai.hx==ai.x){

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

if(ai.hy>ai.y) ai.y=ai.hy-ai.combo-1;

else ai.y=ai.hy+ai.combo+1;

if(ai.y<0 || ai.y >9){

valid=false;

}

if(valid==true){

if(pt[ai.y][ai.x]=='X' || pt[ai.y][ai.x]=='O'){

valid=false;

}

if(pt[ai.y][ai.x]=='O'){

ai.finish=true;

ai.goback=true;

ai.crossdone=true;

ai.combo=0;

}

if(valid==true){

ai.cx=ai.x+48;

ai.cy=ai.y+65;

cout<<"ai fire "<<ai.cy<<ai.cx<<"\n";

if(pt[ai.y][ai.x]!=' '){

pt[ai.y][ai.x]='X';

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

ai.done=true;

ai.combo++;

as->hit++;

}

else{

pt[ai.y][ai.x]='O';

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

ai.done=true;

ai.oneend=true;

ai.oppcombo++;

as->miss++;

}

}

}

else{ //check ->GO TO OPPCOMBO

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

ai.combo=0;

ai.oneend=true;

ai.crossdone=true;

ai.oppcombo++;

ai.combohit=false;

}

}

if(ai.hy==ai.y){

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

if(ai.hx>ai.x) ai.x=ai.hx-ai.combo-1;

else ai.x=ai.hx+ai.combo+1;

if(ai.x<0 || ai.x >9){

valid=false;

ai.combo=0;

ai.goback=true;

ai.finish=true;

}

if(valid==true){

if(pt[ai.y][ai.x]=='X' || pt[ai.y][ai.x]=='O'){

valid=false;

ai.finish=true;

ai.goback=true;

}

if(valid==true){

ai.cx=ai.x+48;

ai.cy=ai.y+65;

cout<<"ai fire "<<ai.cy<<ai.cx<<"\n";

if(pt[ai.y][ai.x]!=' '){

pt[ai.y][ai.x]='X';

ai.combo++;

ai.done=true;

as->hit++;

}

else{

pt[ai.y][ai.x]='O';

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

ai.done=true;

ai.oneend=true;

ai.oppcombo++;

ai.combo=0;

ai.combohit=false;

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

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

as->miss++;

}

}

}

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

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

ai.combo=0;

ai.oneend=true;

ai.crossdone=true;

ai.oppcombo++;

ai.combohit=false;

}

}

}

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

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

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

if(ai.hx==ai.x){

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

if(ai.combohit==false){

if(ai.hy>ai.y) ai.y=ai.hy+ai.oppcombo;

else ai.y=ai.hy-ai.oppcombo;

}

else{

if(ai.y>ai.hy) ai.y=ai.hy+ai.oppcombo;

else ai.y=ai.hy-ai.oppcombo;

}

cout<<ai.y<<ai.x<<endl;

if(ai.y<0 || ai.y>num-1 || pt[ai.y][ai.x]=='O' || pt[ai.y][ai.x]=='X'){

ai.oppcombo=0;

ai.goback=true;

ai.finish=true;

ai.crossdone=true;

ai.combo=0;

ai.done=false;

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

}

else{

ai.cx=ai.x+48;

ai.cy=ai.y+65;

cout<<"ai fire "<<ai.cy<<ai.cx<<"\n";

if(pt[ai.y][ai.x]!=' '){

pt[ai.y][ai.x]='X';

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

ai.done=true;

ai.oppcombo+=1;

ai.combohit=true;

as->hit++;

}

else{

pt[ai.y][ai.x]='O';

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

ai.done=true;

ai.combo=0;

ai.oppcombo=0;

ai.finish=true;

ai.goback=true;

ai.crossdone=true;

as->miss++;

}

}

}

else if(ai.hy==ai.y){

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

if(ai.combohit==false){

if(ai.hx>ai.x) ai.x=ai.hx+ai.oppcombo;

else ai.x=ai.hx-ai.oppcombo;

}

else{

if(ai.x>ai.hx) ai.x=ai.hx+ai.oppcombo;

else ai.x=ai.hx-ai.oppcombo;

}

// cout<<ai.y<<ai.x<<endl;

if(ai.x<0 || ai.x>num-1 || pt[ai.y][ai.x]=='O' || pt[ai.y][ai.x]=='X'){

ai.oppcombo=0;

ai.goback=true;

ai.finish=true;

ai.crossdone=true;

ai.combo=0;

ai.done=false;

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

}

else{

ai.cx=ai.x+48;

ai.cy=ai.y+65;

cout<<"ai fire "<<ai.cy<<ai.cx<<"\n";

if(pt[ai.y][ai.x]!=' '){

pt[ai.y][ai.x]='X';

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

ai.done=true;

ai.oppcombo+=1;

ai.combohit=true;

as->hit++;

}

else{

pt[ai.y][ai.x]='O';

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

ai.done=true;

ai.combo=0;

ai.oppcombo=0;

ai.finish=true;

ai.goback=true;

ai.crossdone=true;

as->miss++;

}

}

}

}

}while(ai.done==false);

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

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

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

ai.over=false;

}

}

if(ai.over==false) turn=1;

else turn=4;

//table

table(pt, fake, real, num);

return turn;

}

void getstates(PlayerS \*ps){

fstream io;

float pttl=0; //total fire

float pac; //accuracy

char pr; //player rank

pttl=ps->hit+ps->miss;

pac=(ps->hit)/pttl;

if(pac>90) pr='S';

else if(pac>80) pr='A';

else if(pac>70) pr='B';

else if(pac>60) pr='C';

else if(pac>50) pr='D';

else if(pac>40) pr='E';

else pr='F';

io.open("accuracy.txt", ios::out | ios::binary);

if(io.fail())

cout<<"Open fail\n";

else{

char m1[]={'R', 'a', 'n', 'k', '='};

io.write(m1, sizeof(m1));

io.write(&pr,sizeof(pac));

io.close();

}

}

//1s delay for ai turn

void pause(){

time\_t start, end; //delay display ai fire

start=time(0);

do{

end=time(0);

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

}

Code – Player.h

/\*

\* File: Player.h

\* Author: Tsz, Kwan

\*

\* Created on September 25, 2014, 11:49 AM

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

struct PlayerT{

char \*\*pt;

};

struct PlayerG{

int unit[5];

int x1, x2, y1, y2;

};

//Status

struct PlayerS{

float hit;

float miss;

};

#endif /\* PLAYER\_H \*/

Code – AI.h

/\*

\* File: AI.h

\* Author: Tsz, Kwan

\*

\* Created on September 30, 2014, 10:34 AM

\*/

#ifndef AI\_H

#define AI\_H

struct AIT{

char \*\*fake, \*\*real;

};

struct AIG{

bool over;

bool done;

bool cross[4];

bool crossdone;

bool goback;

int hx, hy, x, y;

bool finish;

int oppcombo;

int combo;

char cx, cy;

bool hit;

bool oneend;

bool combohit;

};

struct AIS{

float miss, hit;

};

#endif /\* AI\_H \*/