***BATTLESHIP***

**The classic military strategy game**

**Marcus Burton**

**CSC 5**

**Fall 2017**

**47471**

**Size:739 lines**

**Introduction**

“You sunk my battleship!”

-Some kid from the 80s

Battleship. We’ve all played it. The fun guessing game where you try to destroy your enemies ship’s with extreme prejudice.

A 10x10 grid lay on your map, and you must guess where the enemy ships are.

In a world of unfair games, battleship might just be the most fair one you can find.

With only 100 possible positions to lay your ships horizontally or vertically, the game can be quite exciting.

The odds of repeating the same game is 1,000,000,000 to 1!

**Instructions**

The user has five ships, each with their own amount of spaces occupied.

Destroyer with 2 spaces.

Cruisers and Submarines with 3.

Battleships with 4.

And finally, the Carrier with 5.

1. Place down your positions, and be sure to place them next to each other
2. After the ship is placed, the fun begins. Guess the location of the enemies ship before they guess yours.
3. After either the player or AI destroys all the enemies ships, the game is won!

**My approach**

**Making the game work with AI**

I wanted to make the user be able to play against the computer, akin to the old PC games I grew up on(Minesweeper being one of them).

The biggest challenge by far was placing the AI’s ships without overlapping or going out of bounds, but it’s quite simple to declare their limits before the ships are placed!

Though it’s a bit janky, any intelligent user can correctly place their pieces down onto the map correctly, being a good sport to avoid diagonal placing and separating.

It is up to the user to place their ships akin to the rules, much like in the real game. Players who place their ships incorrectly ruin the spirit!

**Similarities to actual board game**

When the player launches an attack and misses, they can see the point so they make sure not to hit it again. And in the same spirit, when a player hits, they mark it down. Same for the AI.

And when a specific type of sink is sunk, the player gets a message of their success sinking X ship, or that X ship of theirs was sunk!

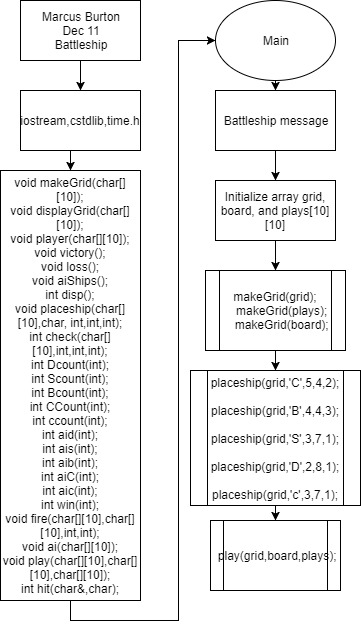
(“You sunk my battleship!”)

**Differences from actual game**

Instead of the victor being chosen when all five ships have been sunk, the winner is chosen when 17 positions have been successfully hit; the amount of pegs in the real life game pieces.

**Logic**

I have a great deal of functions, so I’ll break them down for the sake of simplicity



*Declare function prototypes*

*Print first message*

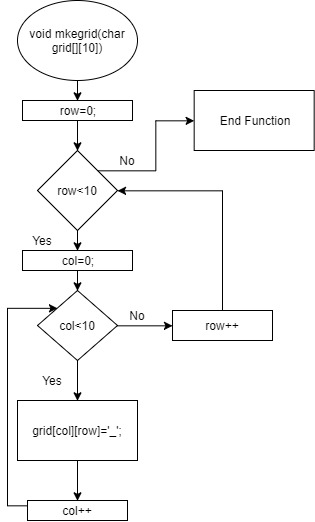
*Declare and initlialize arrays*

*Make the grid for the arrays using makeGrid function*

*Place the ships using placeship functions, passing the grid, name, size, and limits*

*Pass newly made grids to the play function to start the game*

*(Note: this works identically to display grid. Display grid simply uses the same loop to show the board)*

**

*Row = 0*

*If row <10*

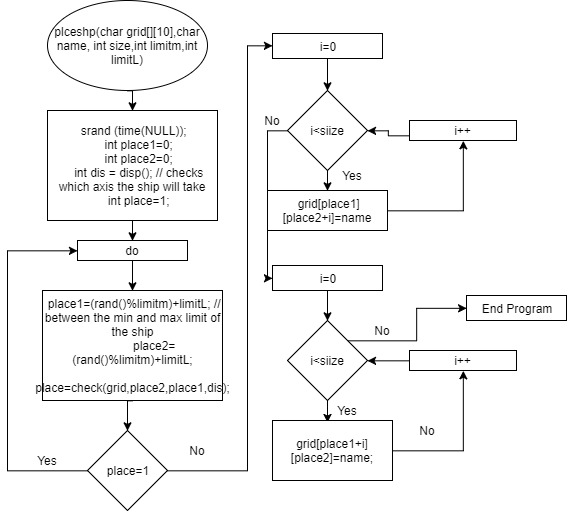
*Col=0*

*If col<10*

*The grid at col and row is empty*

*Add to col*

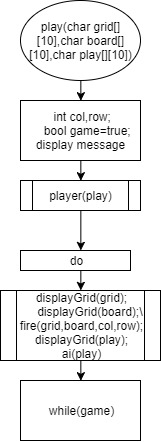
*If col > 9, add to row*

**

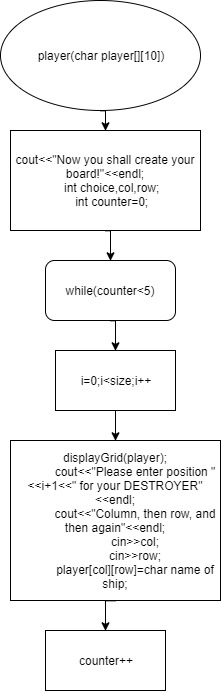
*Get random number generators*

*While the numbers are incorrect for limites, get them again*

*Loop the values into the array*

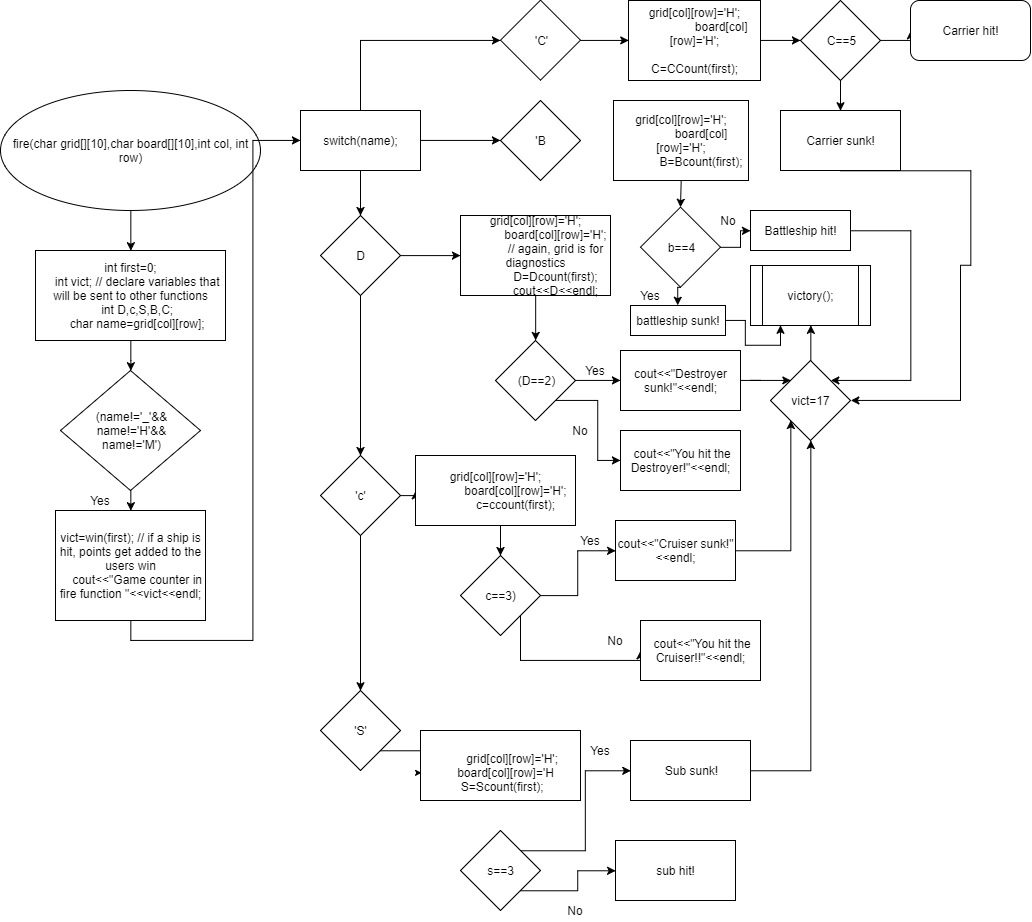
**

*Play game, calling functions and variables in*

**

*Player puts down their marks for each ship*

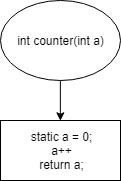
*Loop through 5 ships*

**

*Loop to fire. Included in folder due to blurriness*

*If the random number falls on a ship position, call ship coutner and increment*

*Ai fire function works identically only with different namest*

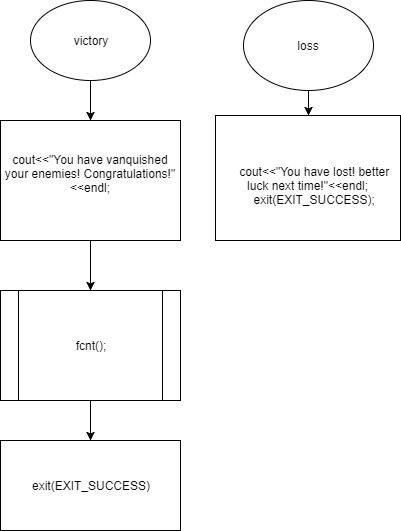
**

*Add to static int*

*Return a*

*Note:this identical counter style is used over and over again for each player hit and ai hit, until a winner is found. Made only one flowchart to reduce redundancy.*

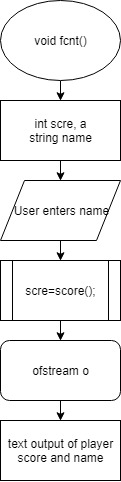
*This style counter usef for:Dcount, Scount, Bcount, CCount, ccount, aid, ais, aib, aiC, aic, win, score, and awin*

**

*Victory and loser functions*

*Display victory or lose message*

*In victory, call funct();*

**

*Initialize variables*

*User enters their name*

*Scre calls score*

*Create file*

*Output score, and plug it into file*

/\*

\* File: main.cpp

\* Author: Marcus Burton

\* Created on December 11th SIZE:32PM

\* Purpose: Battleship!

\*/

//System Libraries Here

#include <iostream>

#include <cstdlib> //random generator

#include <time.h> // for random

#include <fstream> // for file

#include <iomanip>

using namespace std;

//User Libraries Here

const int SIZE = 10;

//Function Prototypes Here

void mkegrid(char[][SIZE]);

void dsplygd(char[][SIZE]);

void player(char[][SIZE]);

void victory();

void loss();

void aiShips();

int disp();

void plceshp(char[][SIZE],char, int,int,int);

int check(char[][SIZE],int,int,int);

int Dcount(int);

int Scount(int);

int Bcount(int);

int CCount(int);

int ccount(int);

int aid(int);

int ais(int);

int aib(int);

int aiC(int);

int aic(int);

int win(int);

void fire(char[][SIZE],char[][SIZE],int,int);

void ai(char[][SIZE]);

void play(char[][SIZE],char[][SIZE],char[][SIZE]);

int hit(char&,char);

int score(int);

void fcnt();

//Program Execution Begins Here

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

cout<<"BATTLESHIP"<<endl;;

char name = '\_';

char grid[SIZE][SIZE];

char board[SIZE][SIZE];

char plays[SIZE][SIZE];

// to check and stuff

mkegrid(grid);

mkegrid(plays);

mkegrid(board);

srand (time(NULL));

//All the ships for the AI

plceshp(grid,'C',5,4,2);

plceshp(grid,'B',4,4,3);

plceshp(grid,'S',3,7,1);

plceshp(grid,'D',2,8,1);

plceshp(grid,'c',3,7,1);

//Uncomment the following to see where the AI ships are located

//cout<<"First, we will show you the AI board for diagnostic purposes"<<endl;

//dsplygd(grid);

play(grid,board,plays);

return 0;

}

void mkegrid(char grid[][SIZE])

{

for(int row=0;row<=SIZE;row++) //loop for rows

{

for(int col=0;col<=SIZE;col++) //loop for columns

{

grid[col][row]='\_';

}

}

}

void dsplygd(char grid[][SIZE])

{

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

for(int row=0;row<SIZE;row++)

{

cout<<row<<" ";

for(int col=0;col<SIZE;col++)

{

cout<<grid[col][row]<<" ";

}

cout<<endl;

}

}

// limitm is limit MAX and limitL is limitLESS(the low end of the limit. it makes no sense I know)

void plceshp(char grid[][SIZE],char name, int size,int limitm,int limitL)

{

int place1=0;

int place2=0;

int dis = disp(); // checks which axis the ship will take

int place=1;

//Uncomment to see the random number that was generated for each ship

//cout<<"ITERATION;"<<dis<<endl;

do

{

place1=(rand()%limitm)+limitL; // between the min and max limit of the ship

place2=(rand()%limitm)+limitL;

place=check(grid,place2,place1,dis);

}while(place==1);

if(dis==1) // checking the iteration

{

for(int i = 0; i <size;i++) // placing all the ships onto the vertical plane

{

grid[place1][place2+i]=name;//vertical

}

}

else if(dis==0)

{

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

{

grid[place1+i][place2]=name; // horizontal

}

}

}

int disp()

{

int xory=0;

xory=rand()%2;//random number between 0 and 1 to find sorting of ships

return xory;

}

int check(char grid[][SIZE],int r, int c, int disp)

{

char check;

if(disp==1) // vertical

{

// checks all the vertical sides

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

{

check=grid[c][i]; //Going through the rows to see if another ship is there

if(check!='\_')

return 1;

}

return 0;

}

else

{

// checks all the horizontal sides

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

{

check=grid[i][r]; //Going through each column to see if another ship

// is in the way

if(check!='\_')

return 1;

}

return 0;

}

}

int hit(char grid[][SIZE],char name) // checking for a hit.

{

for(int col=0;col<SIZE;col++)

{

for(int row=0;row<SIZE;row++)

{

int j = 0;

if(grid[col][row]!=name)

return 1;

else

return 0;

}

}

}

// Checks for a win. if 17 positions are hit that occupy ships, the user wins

int win(int A=0)

{

static int W=A;

W++;

return W;

}

// the following 6 programs count how many times the enemy ship type was hit

int Dcount(int A=0)

{

static int D = 0;

D++;

return D;

}

int Scount(int sub=0)

{

static int S = sub;

S++;

return S;

}

int ccount(int cruise=0)

{

static int c = cruise;

c++;

return c;

}

int Bcount(int Btl=0)

{

static int B=Btl;

B++;

return B;

}

int CCount(int Car=0)

{

static int C = Car;

C++;

return C;

}

int awin(int A=0)

{

static int W=A;

W++;

return W;

}

// The same as above, except for the AI

int aid(int A=0)

{

static int D = 0;

D++;

cout<<D<<endl;

return D;

}

int ais(int sub=0)

{

static int S = sub;

S++;

return S;

}

int aic(int cruise=0)

{

static int c = cruise;

c++;

return c;

}

int aib(int Btl=0)

{

static int B=Btl;

B++;

return B;

}

int aiC(int Car=0)

{

static int C = Car;

C++;

return C;

}

//Firing at the enemy ship

// one of the arrays is for diagnostics, to make sure everything ran fine

void fire(char grid[][SIZE],char board[][SIZE],int col, int row)

{

int first=0;

int vict; // declare variables that will be sent to other functions

int D,c,S,B,C;

char name=grid[col][row];

if(name!='\_'&&name!='H'&&name!='M')

{

vict=win(); // if a ship is hit, points get added to the users win

//cout<<"Game counter in fire function "<<vict<<endl;

}

switch(name)//Each AI ship is located with a letter that signifies their name

// this switch statement figures out which ship was hit

{

case 'D':

{

grid[col][row]='H'; // changes the diagnostic board and player board to H

board[col][row]='H';

// again, grid is for diagnostics

D=Dcount(); // add to Dcount

cout<<D<<endl;

if(D==2)//destroyer as only 2 spaces, if both hit the destroyer is sunk

{

cout<<"Destroyer sunk!"<<endl;

if(vict>=17)// if you hit the enemy 17 times, you

{

victory(); // display victory message if all ships are killed

}

}

else

cout<<"You hit the Destroyer!"<<endl;

break;

}

case 'c': // cruiser

{

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

board[col][row]='H';

c=ccount();

if(c==3){

cout<<"Cruiser sunk!!"<<endl;

if(vict>=17)

{

//if 17 positions are hit, all the ships are sunk, thus a victory

victory();

}

}

else

cout<<"Cruise hit!"<<endl;

break;

}

case 'S'://submarine

{

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

board[col][row]='H';

S=Scount();

if(S==3)

{

cout<<"Submarine sunk!"<<endl;

if(vict>=17)

{

victory();

}

}

else

cout<<"Submarine hit!"<<endl;

break;

}

case 'B'://battleship

{

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

board[col][row]='H';

B=Bcount();

if(B==4)

{

cout<<"You sunk the enemy's battleship!"<<endl;

if(vict>=17)

{

//if 17 positions are hit, all the ships are sunk, thus a victory

victory();

}

}

else

cout<<"Battleship hit!"<<endl;

break;

case 'C':

{

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

board[col][row]='H';

C=CCount();

if(C==5)

{

cout<<"Carrier sunk!"<<endl;

if(vict>=17)

{

//if 17 positions are hit, all the ships are sunk, thus a victory

victory();

}

}

else

cout<<"Carrier hit!"<<endl;

break;

}

case '\_':

{

cout<<"Miss!"<<endl;

board[col][row]='M';

break;

}

case 'H':

{

cout<<"You already hit that target!!!"<<endl;

break;

}

}

}

}

// the function that plays the actual game. it calls other functions mainly

void play(char grid[][SIZE],char board[][SIZE],char play[][SIZE])

{

int col,row;

int Scre;

int chk=0;

bool game=true;

// the user places their ships

cout<<"First, place down your positions onto the board!"<<endl;

player(play);

do

{

// This loop goes through the game until a victor is chosen.

// the game value doesn't change on purpose

dsplygd(board); //Plain board

cout<<"Where would you like to fire! Enter the Column, then the row"<<endl;

cin>>col;

cin>>row;

fire(grid,board,col,row);// call fire function to check the hit

Scre=score(chk);//counting up the score. each fire adds to the score.

dsplygd(play);//display the grid

cout<<"Now the AI shall take their shot!"<<endl;

ai(play);

}while(game); // the loop is always true, so the game continues until

// a victor is chosen

}

void victory()

{

// and here is your generic victory message

cout<<"You have vanquished your enemies! Congratulations!"<<endl;

fcnt();

exit(EXIT\_SUCCESS); // program exits

}

void loss()

{

//Generic loss message

cout<<"You have lost! better luck next time!"<<endl;

exit(EXIT\_SUCCESS); // program exits

}

void player(char player[][SIZE])

{

int choice,col,row;

int counter=0;

bool acct=false;

// Loop goes through every ship(There's five);

while(counter<5)

{

for(int i =0;i<2;i++) // several for loops go through the ship sizes and place the markers down, changing the grid for the AI.

{

dsplygd(player);

cout<<"Please enter position "<<i+1<<" for your DESTROYER"<<endl;

cin>>col;

cin>>row;

while(col <0||col>9 || row<0 || row>9 || player[col][row]!='\_')

{

cout<<"Incorrect placement! Try again."<<endl;

cout<<"Please enter position "<<i+1<<" for your DESTROYER"<<endl;

cin>>col;

cin>>row;

}

player[col][row]='D';

}

counter++;

dsplygd(player);

for(int i = 0;i<3;i++)

{

cout<<"Please enter position "<<i+1<<" for your SUB. same as before!"<<endl;

cin>>col;

cin>>row;

while(col <0||col>=SIZE || row<0 || row>=SIZE || player[col][row]!='\_')

{

cout<<"Incorrect placement! Try again."<<endl;

cout<<"Please enter position "<<i+1<<" for your SUB. same as before!"<<endl;

cin>>col;

cin>>row;

}

player[col][row]='S';

dsplygd(player);

}

counter++;

for(int i=0;i<3;i++)

{

cout<<"Please enter position "<<i+1<<" for your cruiser"<<endl;

cin>>col;

cin>>row;

while(col <0||col>=SIZE || row<0 || row>=SIZE || player[col][row]!='\_')

{

cout<<"Incorrect placement! Try again."<<endl;

cout<<"Please enter position"<<i+1<<" for your cruiser"<<endl;

cin>>col;

cin>>row;

}

player[col][row]='c';

dsplygd(player);

}

counter++;

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

{

cout<<"Please enter position "<<i+1<<" for your BATTLESHIP!"<<endl;

cin>>col;

cin>>row;

while(col <0||col>=SIZE || row<0 || row>=SIZE || player[col][row]!='\_')

{

cout<<"Incorrect placement! Try again."<<endl;

cout<<"Please enter position "<<i+1<<" for your BATTLESHIP!"<<endl;

cin>>col;

cin>>row;

}

player[col][row]='B';

dsplygd(player);

}

counter++;

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

{

cout<<"Please enter position "<<i+1<<" for your CARRIER!"<<endl;

cin>>col;

cin>>row;

while(col <0||col>=SIZE || row<0 || row>=SIZE || player[col][row]!='\_')

{

cout<<"Incorrect placement! Try again."<<endl;

cout<<"Please enter position "<<i+1<<" for your CARRIER!"<<endl;

cin>>col;

cin>>row;

}

player[col][row]='C';

dsplygd(player);

}

counter++;

}

}

void ai(char player[][SIZE]) // this function is the AI attacking the players ships

{

int col = rand()%SIZE; // first declare the locations on the function

int row=rand()%SIZE;

int first=0;

int vict;

int D,c,S,B,C;

char name=player[col][row];// calls the name of the ship on the locale, or it can be a miss or a re-hit.

do

{

// the AI picks a random position to fire upon

col=rand()%SIZE;

row=rand()%SIZE;

}while(player[col][row]=='H'||player[col][row]=='M'||player[col][row]!='\_'); // the AI picks a new location if they fired upon a specific position before

//Obviously any player wouldn't hit the same place twice, so I threw in this logic that makes the computer not make that mistake, to make the game more challenging.

if(name!='\_'&&name!='H'&&name!='M')

{

vict=awin(first); // every time a hit lands, the counter goes up. Max being 18.

// cout<<"Game counter in fire function "<<vict<<endl;

}

switch(name)

{

// same loop as for the player, but situated for the AI's function counters.

case 'D':

{

player[col][row]='H';

D=aid();

cout<<D<<endl;

if(D==2)

{

cout<<"Our Destroyer was sunk!"<<endl;

if(vict>=17)

{

loss();

}

}

else

cout<<"You hit the Destroyer!"<<endl;

break;

}

case 'c':

{

player[col][row]='H';

c=aic();

if(c==3){

cout<<" Our Cruiser sunk!!"<<endl;

if(vict>=17)

{

loss();

}

}

else

cout<<"Our Cruiser hit!"<<endl;

break;

}

case 'S':

{

player[col][row]='H';

S=ais();

if(S==3)

{

cout<<"Our Submarine sunk!"<<endl;

if(vict>=17)

{

loss();

}

}

else

cout<<"Our Submarine hit!"<<endl;

break;

}

case 'B':

{

player[col][row]='H';

B=aib();

if(B==4)

{

cout<<"They sunk our battleship!"<<endl;

if(vict>=17)

{

loss();

}

}

else

cout<<"Battleship hit!"<<endl;

break;

case 'C':

{

player[col][row]='H';

C=CCount(first);

if(C==5)

{

cout<<"Carrier sunk!"<<endl;

if(vict>=17)

{

loss();

}

}

else

cout<<"Carrier hit!"<<endl;

break;

}

case '\_':

{

cout<<"Miss!"<<endl;

player[col][row]='M';

break;

}

case 'H':

{

cout<<"They already hit that target!!!"<<endl;

break;

}

}

}

}

int score(int C=0) // function counts up the players score

{

static int S=C;

S++;

return S;

}

void fcnt() // this prints out the players name and score!

{

int scre = score();

int a; // pulled from txt

cout<<endl;

ofstream o;// prints onto file

int winner;

cout<<"Please enter your name, oh victorious one!"<<endl;

string name;

cin>>name;

cout<<"NAME"<<setw(10)<<"SCORE"<<endl;

cout<<name<<setw(10)<<scre<<endl;

o<<"NAME"<<setw(10)<<"SCORE"<<endl;

o<<name<<setw(10)<<scre<<endl;

o.close();

}