**Project 1**

( Yut Nori Game )

CIS 5

KyungJin Shin

**Table of Contents**

Introduction

Rules of the Game

Summary of your development

Example Inputs with Outputs

Flowcharts

Pseudo-code

Completed Check-off Sheet

1. **Introduction**

Yut Nori Game is the traditional Korean New year season board game. The game is played by people of all ages. 2 players can play the game and each one has one board piece to play the game. In this program, computer and user will play the game. This game starts with Rock-Scissor-Paper to decide who goes first. If user wins, user will throw the dice (4 sticks) for Yut Nori Game. If user loses, computer will throw the dice (4 sticks) for Yut Nori Game. This game is simple throwing 4 sticks to move their board piece, but it is also competitive because each player has a lot of chance to catch up (ahead of ) the other player’s game piece. Both computer’s piece (y) and user’s piece (x) are starting from ( 0, 0 ) position. Arriving first at the end point (0, 0) wins the game (finish 1 lap). When user or computer passes the end position, the game will stop, display who was win, and file the winner. I chose this game because anyone can learn the game quickly and enjoy it.

1. **Rules of the Game**

2 players can play this game. If user goes first, computer is the next. Their turn is alternative, and they can throw the 4 wood sticks each time on their turns. User’s board piece is (x) and computer’s board piece is (y). They start at the same position which is (0, 0) and go around the board until one player reaches the arriving position at (0, 0) first is the winning the game for this Project.

There are 4 wood sticks to determine how much the player’s piece move. It is similar as flipping a coin. Wood sticks’ front side is black, and back side is white.

Example)

* If player gets ( black, white, white, white ) or ( white, black, white, white )

or ( white, white, black, white ) or ( white, white, white, black )

1 black and 3 white → Move 1 step.

* If player gets ( black, black, white, white ) or ( black, white, black, white )

or ( black, white, white, black ) or ( white, black, black, white )

or ( white, black, white, black ) or ( white, white, black, black )

2 black and 2 white → Move 2 steps.

* If player gets ( black, black, black, white ) or ( black, black, white, black )

or ( black, white, black, black ) or ( white, black, black, black )

3 black and 1 white → Move 3 steps.

* If player gets ( black, black, black, black )

4 black and 0 white → Move 4 steps, and player has bonus to throw the sticks again

* If I get ( white, white, white, white )

0 black and 4 white → Move 5 steps, and player has bonus to throw the sticks again

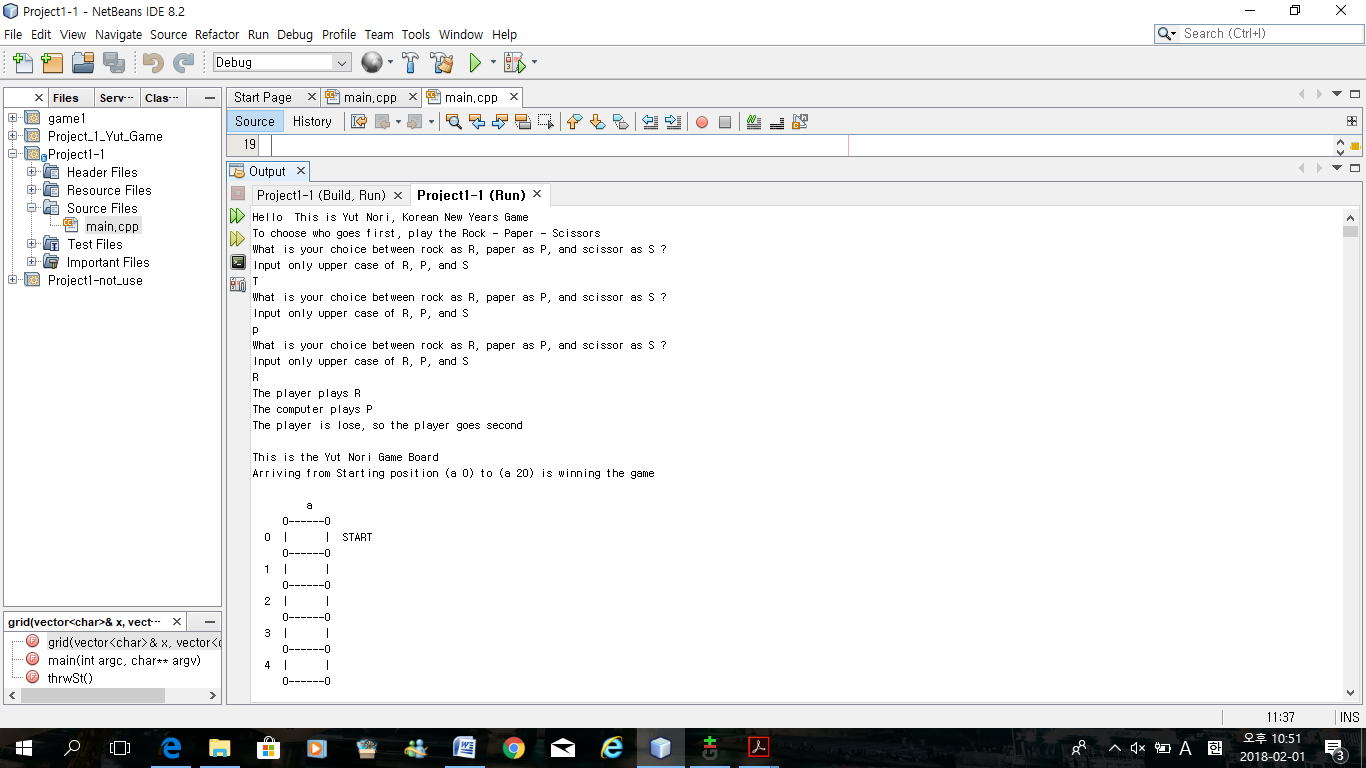
If user and computer are on the same position, second arrived piece could catch the first arrived piece, so second arrived piece will take the place, and the first arrived piece has to go back to the starting point.

1. **Summary of your development**

I chose this game because I had been playing this game with my family since I was 4 years old. Programming this game was difficult than I thought. First of all, I was a struggle to display the board grid. Moving around the square corner requires making a lot of columns and rows. It was very long code and not working when I made the code without arrays and function. However, when I tried arrays and pair (Michael introduced) it was working, so I decide to use several arrays and pair to determine coordinate the each board position. In addition, I made for loops each one for user and computer to throw 4 wood sticks and display their pieces positions. Unfortunately, it was working only one loop. I wanted alternative turns, but it was working only user’s turns and finished the game, without computer’s turn. I had to figure it out how to make alternative turns. I tried to combine user’s for loop and computer’s for loop into one for loop. Also, I tried to make one big for loop and two for loops inside the one big for loop. They were still not working. Finally one big do while loop and two for loops inside the one big do while loop was working. After all, sometimes I got the result successfully, and sometimes the result was failed, although they are same code. I have to block the code and check each line to figure it out what is the problem. I think locating delete the structure was the problem.

1. **Example Inputs with Outputs**

If I do not input upper case of R,P, and S, It will loop until I input the right one (R or P or S).



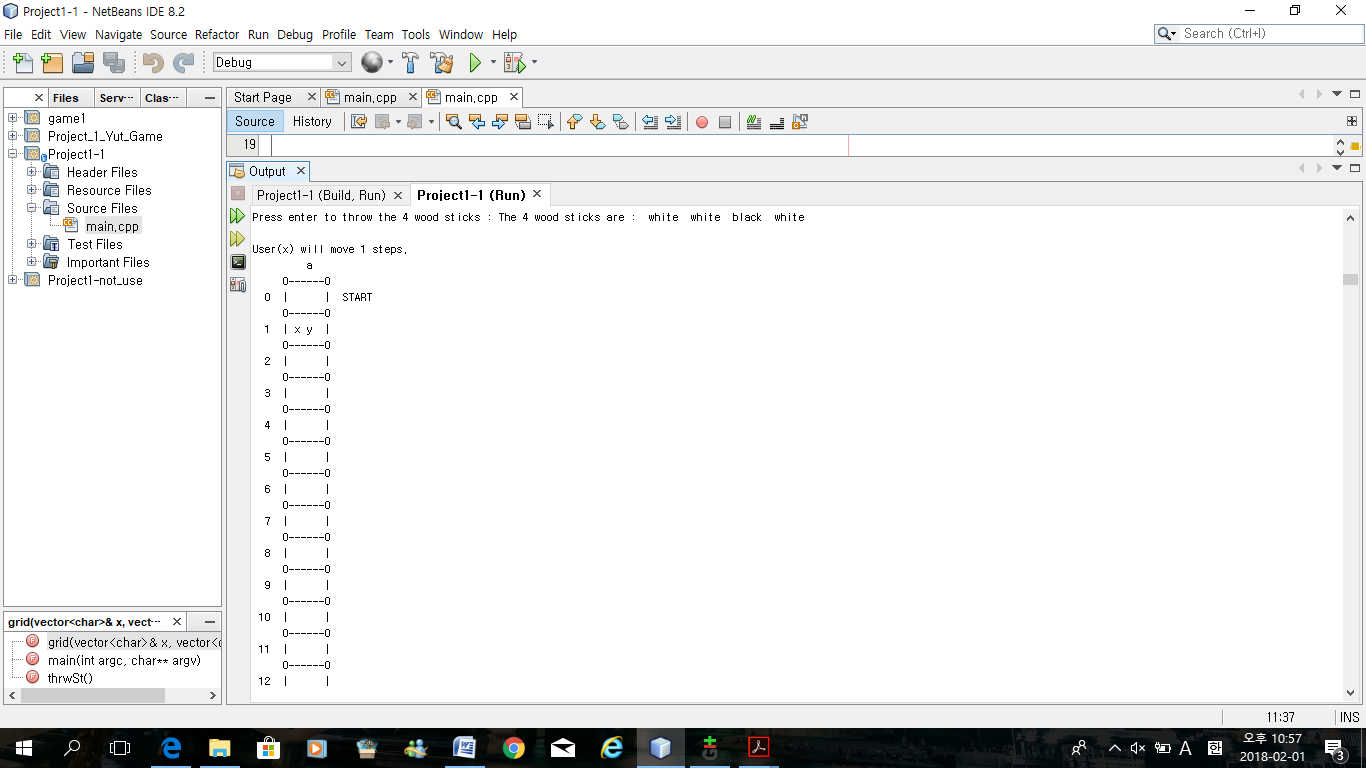
This is the possible percentage when you throw the 4 sticks each time.

Because the player loses the Rock – Scissor – Paper game, Computer will go first.

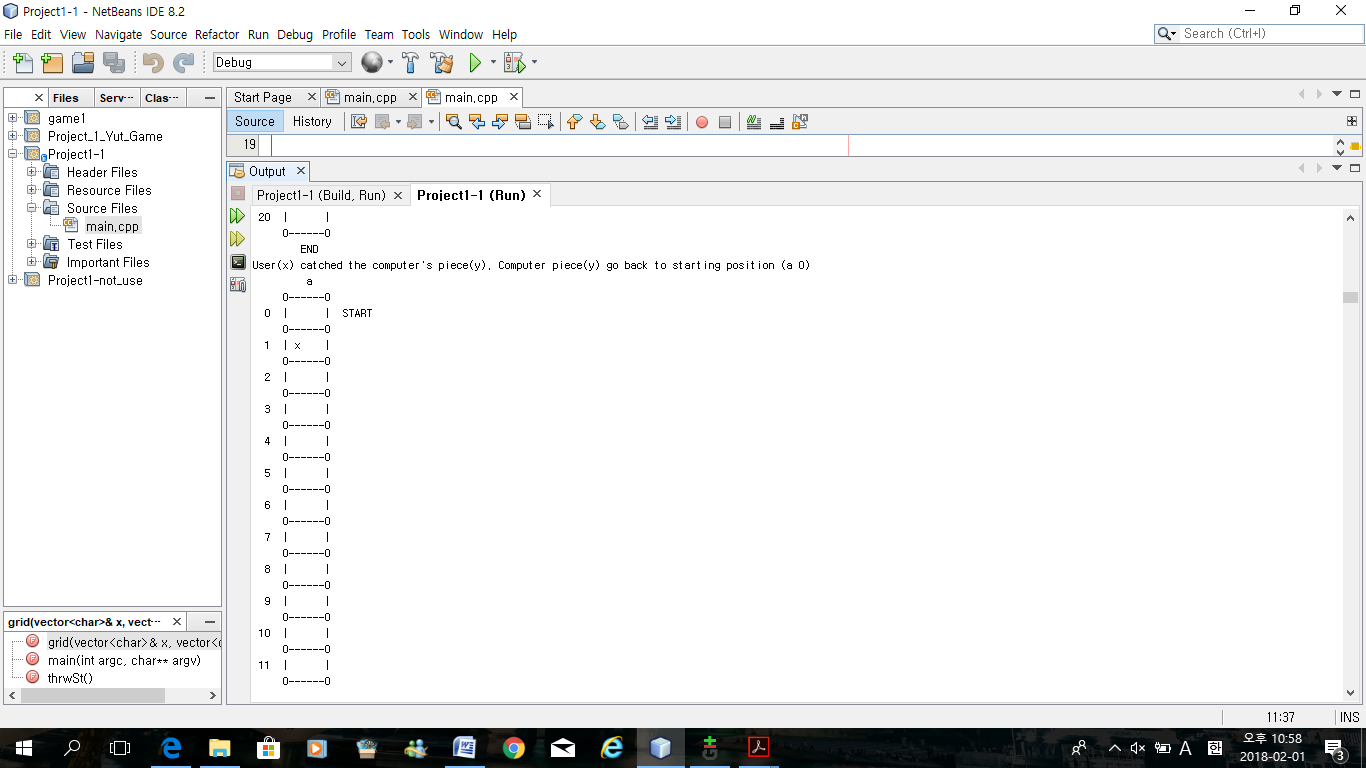
Computer got 1 black and 3 white, so computer will moves 1 steps.

It displays as a (y) on the board grid.

If computer and user are on the same position,

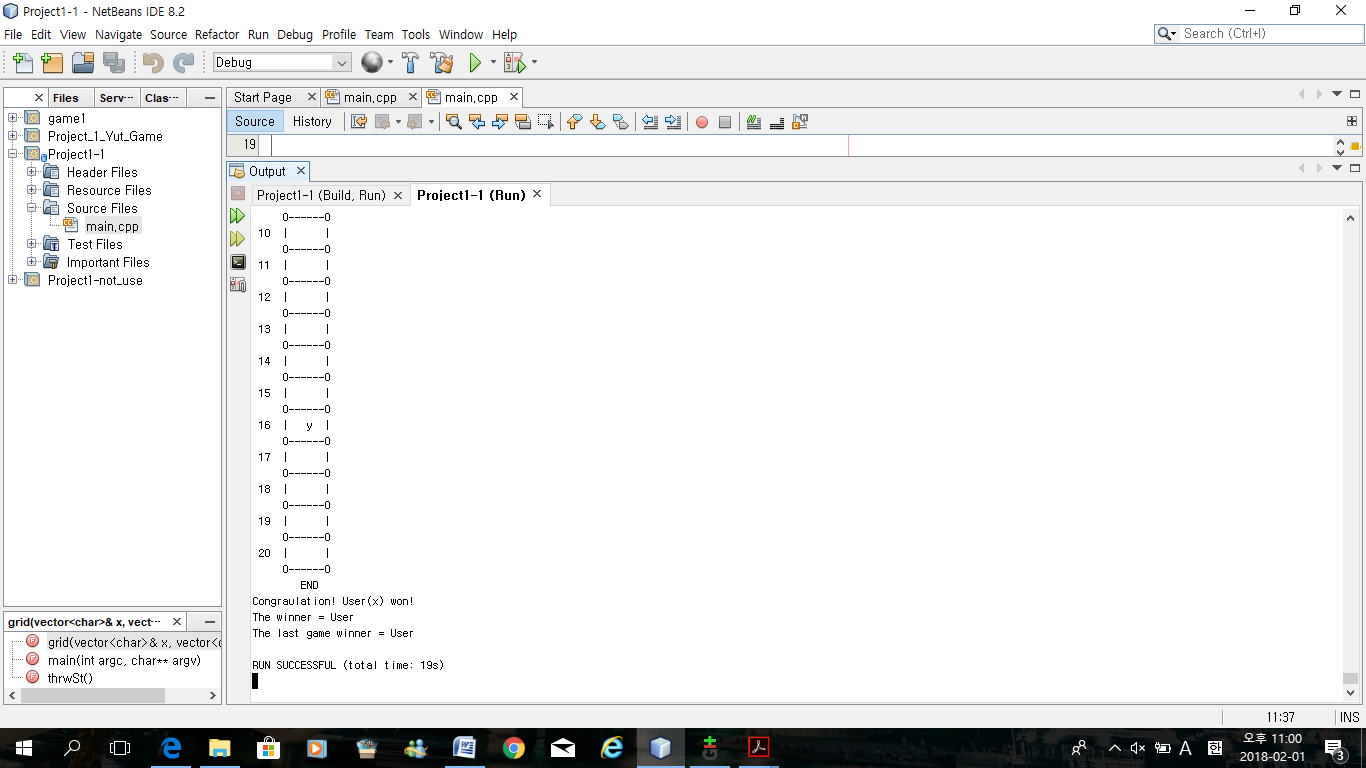


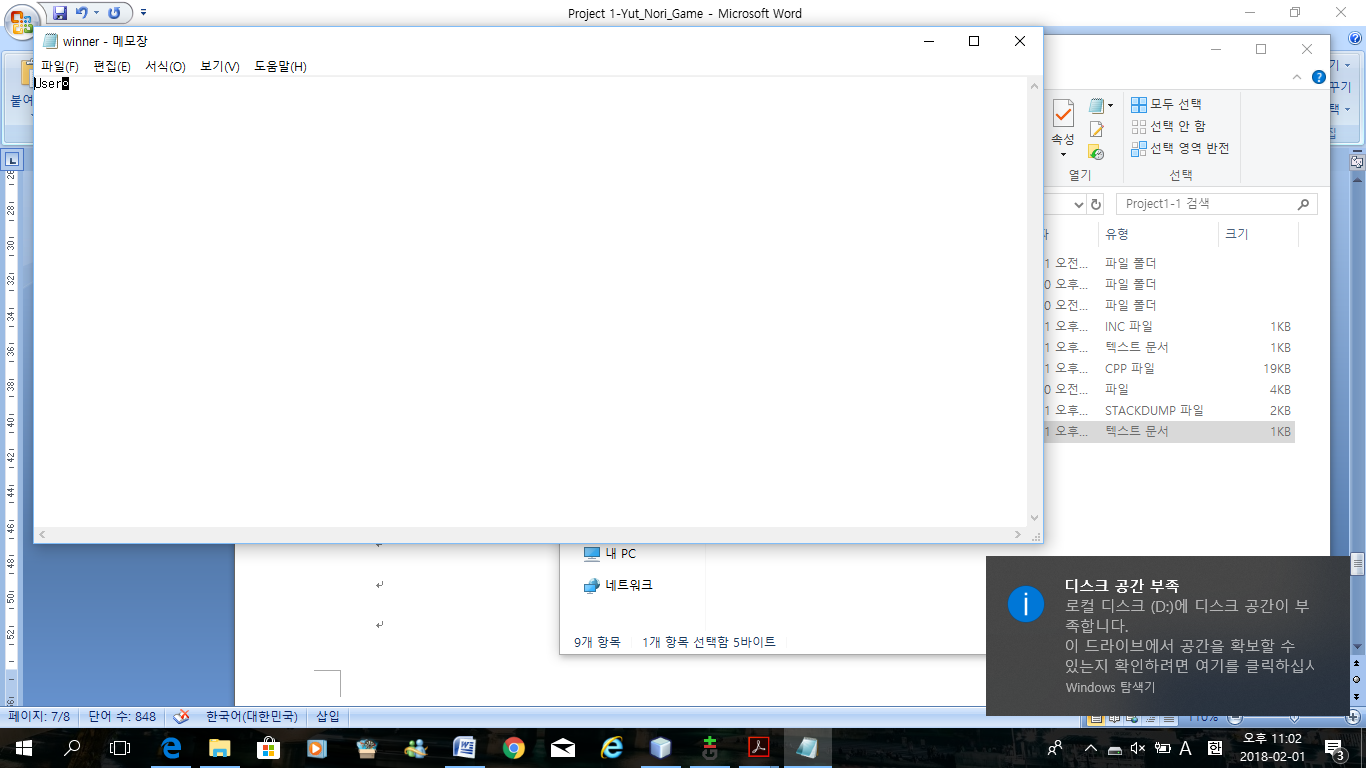
User’s piece (x) catches the computer piece’s (y). Therefore, computer has to go back to starting position.

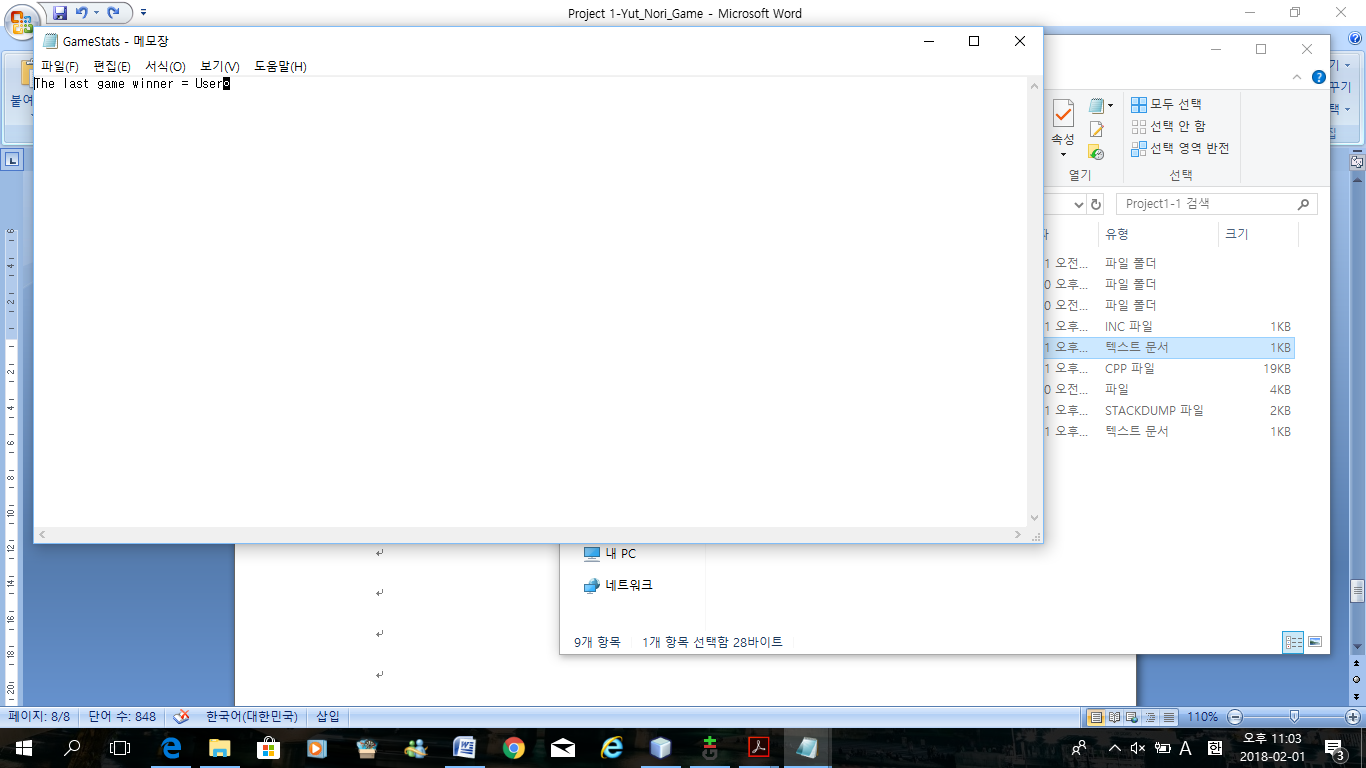


Finally, when user passed the ( a20 ) position first, user won the game.

It will shows coagulation sign and it will file who is the winner.







1. **Flowcharts**

When I attached the flowchart jpg, it was not clear, so I attach separately.

1. **Pseudo-code**

//system Libraries

#include <iostream> //I/O Library

#include <cmath> //Math Library

#include <iomanip> //Formatting Library

#include <cstdlib> //Random Number Generator

#include <ctime> //Time to Seed Random Number

#include <vector> //Vector Function Library

#include <fstream> //File Library

#include <string> //String Library

using namespace std;

//User Libraries

//Global Constants - Math/Physics Constants, Conversions,

// 2-D Array Dimensions

**//Function Prototypes**

void grid(vector<char> &, vector<char> &);

int thrwSt();

**//Execution Begins Here**

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

**//Generate Random Number Seed**

srand(static\_cast<unsigned int>(time(0))); //To get Random Number Generator

**//Declare Variables**

char comput; //Computer for Rock-Paper-Scissors game

char player; //Player for Rock-Paper-Scissors game

short user=0; //User for Yut Nori Game

short comp=0; //Computer for Yut Nori Game

int num; //Integer Number, but I will range to 0 to 3

string winner; //To file the winner of Yut Nori Game

**//Variables for wood sticks game**

vector<char> x(21, ' '); // to store 'x' on a current position and a space on other positions

vector<char> y(21, ' '); // to store 'y' on a current position and a space on other positions

**//Declare file and game variables**

ofstream out; //Output File

fstream data; //Input and Output File

data.open("winner.txt", ios::out); //Open the output file

**//Initialize variables**

char outName[]="GameStats.txt"; //Character Array Name

out.open(outName); //Open the Output File

**//Start the Game**

string greet = "Hello "; //Print out "Hello" when I use greet

cout<<greet<<" This is Yut Nori, Korean New Years Game"<<endl;

cout<<"To choose who goes first, play the Rock - Paper - Scissors"<<endl;

**//Rock - Paper - Scissors Game**

//Initialize Variables

num=rand()%3; // number of 0 to 2

switch(num){

case 0: comput='R'; break; //when number is 0, computer print out 'R'

case 1: comput='S'; break; //when number is 1, computer print out 'S'

case 2: comput='P'; break; //when number is 2, computer print out 'P'

}

while(player!='R' && player!='S' && player!='P') {

cout<<"What is your choice between rock as R, paper as P, and scissor as S ?"<<endl;

cout<<"Input only upper case of R, P, and S"<<endl;

cin>>player; //If you input other than upper case of R,S,P, you have to try again.

}

**//Process/Map inputs to outputs**

cout<<"The player plays "<<player<<endl; //Display what player chose

cout<<"The computer plays "<<comput<<endl; //Display what computer chose

if(comput==player) { //When computer and player chose same thing

cout<<"This game is tie"<<endl;

do{

cout<<"What is your choice between rock as R, paper as P,”

<< “and scissor as S ?"<<endl;

cin>>player;

cout<<"The player plays "<<player<<endl;

cout<<"The computer plays "<<comput<<endl;

}while(comput==player); //Loop this until computer and player chose different

}

**/\*------------------------Player goes First ------Yut Nori Game--------------------------------\*/**

if((comput=='R' && player=='P') || (comput=='P' && player=='S')

|| (comput=='S' && player=='R')){

cout<<"The player is win, so the player goes first"<<endl;

//When player choose Paper,and computer choose Rock

//When player choose Scissor,and computer choose Paper

//When player choose Rock,and computer choose Scissor

**/\*------------------------Yut Nori Game-------------------------------------------------\*/**

cout<<endl<<"This is the Yut Nori Game Board"<<endl;

cout<<"Arriving from Starting position (a 0) to (a 20) is winning”

<< “the game"<<endl<<endl;

grid(x,y); //Display the Yut Nori Board Game Grid

float poss,a,b,c; //Possible, 3 different chances to get when you throw the sticks

a=0.25f; //1/4 chance to get (1 black,3 white) or (4 black,1 white)

b=0.375f; //3/8 chance to get (2 black,2 white)

c=0.0625f; //1/16 chance to get (4 black,0 white) or (0 black,4 white)

float max; //Find maximum chances

poss = c<a?max:0;

max=fmax(a,c); //Using (cmath Library) to determine which one is more

// possible to get frequently between a and c

cout<<fixed<<setprecision(2)<<showpoint; //2 decimal points

cout<<endl<<"When you throw the 4 sticks, "<<endl;

cout<<"Percentage to get 1 black and 3 white = "

<<a\*100.0f<<"%"<<endl; //(1/4)\*100% possible

cout<<"Percentage to get 2 black and 2 white = "

<<b\*100.0f<<"%"<<endl; //(3/8)\*100% possible

cout<<"Percentage to get 3 black and 1 white = "

<<max\*100.0f<<"%"<<endl; //(1/4)\*100% possible

cout<<"Percentage to get 4 black and 0 white = "

<<setw(5)<<c\*100.0f<<"%"<<endl; //(1/16)\*100% possible

cout<<"Percentage to get 0 black and 4 white = "

<<setw(5)<<c\*100.0f<<"%"<<endl<<endl; //(1/16)\*100% possible

comp=0; //Initialize computer is 0

user=0; //Initialize user is 0

int count=0; //Initialize count is 0

int nsteps = 21; //Total positions are 21 steps

cout<<"User(x),Computer(y)"<<endl; //User's piece is x, Computer's piece is

//y on the board grid

do{

**//Player goes first**

cout<<"Press enter to throw the 4 wood sticks : ";

cin.get(); //Enter to move on

do {

count = thrwSt(); x[user]=' ';

//make x[position]=' '; somewhere after you display it and before you throw sticks again

user += count; //User's positions adding up from each count

cout<<endl<<"User(x) will move "<<count<<" steps."<<endl;

//Explain how much user will move from the result of sticks

**//Print the User's Grid**

x[user]='x'; // mark 'x'

grid(x,y); //Show marks on the grid

if (user==comp){

y[comp]=' '; //mark ' '. Previous mark will disappear

count=0;//Initialize count is 0

comp=0; //Initialize computer is 0. Computer go back to starting

//position because user's piece catched the computer's piece

cout<<"User(x) catched the computer's piece(y). Computer piece(y)”

“go back to starting position (a 0)"<<endl;

grid(x,y); //Show new marks on the grid

y[comp]='y'; //mark 'y'

cout<<endl;

if (count==4 || count==5 ) cout<<"User(x) has bonus to throw the sticks”

<<” again"<<endl;

} //Only when sticks are white,white,white,white or

//black,black,black,black, throw one more time

}while(count==4 || count==5); //Loops until sticks are white,white,white,white or

//black,black,black,black

**//Computer goes next**

cout<<"And now Computer's(y) turn..."<<endl<<endl;

do{

count = thrwSt(); y[comp]=' '; //make y[position]=' '; somewhere after

//you display it and before you throw sticks again

**//Print the Computer's Grid**

if (count==0) count=5; //If you get 4 white, you will move 5 steps

cout<<endl<<"Computer(y) will move "<<count<<" steps."<<endl; //Explain how much user will move from the result of sticks

comp+=count; //Computer's positions adding up from each count

y[comp]='y'; // mark 'y'

grid(x,y); //Display the new grid

if (comp==user){

x[comp]=' '; //mark ' ', previous mark was disappeared

count=0;//Initialize count is 0

user=0;//Initialize user is 0. User go back to starting position

//because computer's piece catched the user's piece

cout<<"Computer(y) catched the user's piece(x). User piece(x) go”

<<” back to starting position (a 0)"<<endl;

grid(x,y); //Show new marks on the grid

x[user]='x'; //mark 'x'

cout<<endl;

if (count==4 || count==5 ) cout<<"Computer(y) has bonus to throw the”

<< “sticks again"<<endl;

}

}while(count==4 || count==5); //Loops until sticks are

//white,white,white,white or black,black,black,black

if (user>=nsteps){ //When user passed the a20 position

cout<<"Congraulation! User(x) won!"<<endl;

data<<"User"<<endl; //When user won, File the "User"

}else if (comp>=nsteps){ //When computer passed the a20 position

cout<<"Sorry. Computer(y) has won. User(x) lost"<<endl;

data<<"Computer"<<endl; //When computer won, File the "Computer"

}

}while(!(comp>=nsteps) && !(user>=nsteps)); //Loops stop when computer piece or

// user piece arrived at end of the position

//same as = while(!(comp>=nsteps) || !(user>=nsteps))

//same as = while(comp<nsteps || user<nsteps)

**/\*------------------------Computer goes First ------Yut Nori Game--------------------------------\*/**

}else if((comput=='R' && player=='S') || (comput=='P' && player=='R') || (comput=='S' && player=='P')) {

cout<<"The player is lose, so the player goes second"<<endl;

//When player choose Scissor,and computer choose Rock

//When player choose Rock,and computer choose Paper

//When player choose Paper,and computer choose Scissor

**/\*----------------------------------Yut Nori Game--------------------------------------------\*/**

cout<<endl<<"This is the Yut Nori Game Board"<<endl;

cout<<"Arriving from Starting position (a 0) to (a 20) is winning the game"<<endl<<endl;

grid(x,y); //Display the Yut Nori Board Game Grid

float a,b,c; //3 different chances to get when you throw the sticks

a=0.25f; //1/4 chance to get (1 black,3 white) or (4 black,1 white)

b=0.375f; //3/8 chance to get (2 black,2 white)

c=0.0625f; //1/16 chance to get (4 black,0 white) or (0 black,4 white)

float max; //Find maximum chances

max=fmax(a,c); //Using (cmath Library) to determine which one is more possible to get frequently between a and c

cout<<fixed<<setprecision(2)<<showpoint; //2 decimal points

cout<<endl<<"When you throw the 4 sticks, "<<endl;

cout<<"Percentage of getting 1 black and 3 white = "

<<a\*100.0f<<"%"<<endl; //(1/4)\*100% possible

cout<<"Percentage of getting 2 black and 2 white = "

<<b\*100.0f<<"%"<<endl; //(3/8)\*100% possible

cout<<"Percentage of getting 3 black and 1 white = "

<<max\*100.0f<<"%"<<endl; //(1/4)\*100% possible

cout<<"Percentage of getting 4 black and 0 white = "

<<setw(5)<<c\*100.0f<<"%"<<endl; //(1/16)\*100% possible

cout<<"Percentage of getting 0 black and 4 white = "

<<setw(5)<<c\*100.0f<<"%"<<endl<<endl; //(1/16)\*100% possible

comp=0; //Initialize computer is 0

user=0; //Initialize user is 0

int count=0; //Initialize count is 0

int nsteps = 21; //Total positions are 21 steps

cout<<"User(x),Computer(y)"<<endl; //User's piece is x, Computer's piece is y on the board grid

do{

//Computer goes first

cout<<"And now Computer's(y) turn..."<<endl<<endl;

do{

count = thrwSt(); y[comp]=' '; //make y[position]=' '; somewhere after you display it and before you throw sticks again

**//Print the Computer's Grid**

if (count==0) count=5; //If you get 4 white, you will move 5 steps

cout<<endl<<"Computer(y) will move "<<count<<" steps."<<endl; //Explain how much user will move from the result of sticks

comp+=count; //Computer's positions adding up from each count

y[comp]='y'; // mark 'y'

grid(x,y); //Display the new grid

if (comp==user){

x[comp]=' '; //mark ' ', previous mark was disappeared

count=0;//Initialize count is 0

user=0;//Initialize user is 0. User go back to starting position

//because computer's piece catched the user's piece

cout<<"Computer(y) catched the user's piece(x). User piece(x) go back to”

<< “starting position (a 0)"<<endl;

grid(x,y); //Show new marks on the grid

x[user]='x'; //mark 'x'

cout<<endl;

if (count==4 || count==5 ) cout<<"Computer(y) has bonus to throw the sticks”

<< “again"<<endl;

}

}while(count==4 || count==5); //Loops until sticks are white,white,white,white or

//black,black,black,black

**//Player goes first**

cout<<"Press enter to throw the 4 wood sticks : ";

cin.get(); //Enter to move on

do {

count = thrwSt(); x[user]=' '; //make x[position]=' '; somewhere after you display it and before you throw sticks again

user += count; //User's positions adding up from each count

cout<<endl<<"User(x) will move "<<count<<" steps."<<endl;

//Explain how much user will move from the result of sticks

**//Print the User's Grid**

x[user]='x'; // mark 'x'

grid(x,y); //Show marks on the grid

if (user==comp){

y[comp]=' '; //mark ' '. Previous mark will disappear

count=0;//Initialize count is 0

comp=0; //Initialize computer is 0. Computer go back to starting

//position because user's piece catched the computer's piece

cout<<"User(x) catched the computer's piece(y). Computer piece(y) go”

<< “back to starting position (a 0)"<<endl;

grid(x,y); //Show new marks on the grid

y[comp]='y'; //mark 'y'

cout<<endl;

if (count==4 || count==5 ) cout<<"User(x) has bonus to throw the sticks”

<<”again"<<endl;

} //Only when sticks are white,white,white,white or

//black,black,black,black, throw one more time

}while(count==4 || count==5); //Loops until sticks are

//white,white,white,white or black,black,black,black

if (user>=nsteps){ //When user passed the a20 position

cout<<"Congraulation! User(x) won!"<<endl;

data<<"User"<<endl; //When user won, File the "User"

}else if (comp>=nsteps){ //When computer passed the a20 position

cout<<"Sorry. Computer(y) has won. User(x) lost"<<endl;

data<<"Computer"<<endl; //When computer won, File the "Computer"

}

}while(!(comp>=nsteps) && !(user>=nsteps)); //Loops stop when computer piece or user piece arrived at end of the position

//same as = while(!(comp>=nsteps) || !(user>=nsteps))

// same as = while(comp<nsteps || user<nsteps)

}

**//Output the game statistics to the screen**

data.close(); //Close the Output Data File

data.open("winner.txt", ios::in); //Open the Input Data File

if (data>>winner){

cout<<"The winner = "<<winner<<endl; //File the winner

}

cout<<"The last game winner = "<<winner<<endl; //Display on the screen

**//Output the game statistics to a file**

if (data>>winner){

cout<<"The winner = "<<winner<<endl; //File the winner

}

out<<"The last game winner = "<<winner<<endl; //Display on the GameStats.txt File

//Close Files and Exit stage right!

data.close();

out.close();

return 0;

}

**int thrwSt()** {

/\*

true | false

true == black

false = white

\*/

int count = 0; //Initialize count is 0

bool sticks[4] = {false, false, false, false}; //Order of white or black does not matter. Total black and total white is matter.

cout<<"The 4 wood sticks are :"<< setw(7); // start displaying a message

for( int i = 0; i<4; i++ ){ //When position is 0

sticks[i]=rand()%2; // 0 (false) or 1 (true)

cout<<setw(7); // so "black" or "white" is shown with 7 width

if( sticks[i] ){

//its black

count++; //Incresement each time

cout <<"black"; //True for black

} else cout << "white"; //False for white

}

if (count==0) count=5; //If you get 4 white, you will move 5 steps

cout<<endl;

return count; //Return to count

}

**void grid(vector<char> &x, vector<char> &y)** {

cout<<setw(10)<<"a"<<endl;

cout<<setw(13)<<right<<"0------0"<<endl;

for(short i=0; i<1; i++) { // Display a0 position

cout<<setw(3)<<i<<" | "<<x[i]<<" "<<y[i]<<" | START"<<endl;

cout<<setw(13)<<"0------0"<<endl;

}

for(short i=1; i<20; i++) { // Display a1 to a19 positions

cout<<setw(3)<<i<<" | "<<x[i]<<" "<<y[i]<<" |"<<endl;

//user piece(x) and computer piece(y) will show in the square position

cout<<setw(13)<<"0------0"<<endl;

}

for(short i=20; i<21; i++) { // Display a20 position

cout<<setw(3)<<i<<" | "<<x[i]<<" "<<y[i]<<" |"<<endl;

//user piece(x) and computer piece(y) will show in the square position

cout<<setw(13)<<"0------0"<<endl;

cout<<setw(13)<<" END "<<endl;

}

}

1. **Completed Check-off Sheet**

When I attached the Check off sheet, it was not clear, so I attach separately.