**Project 1**

( Yut Nori Game )

CIS 5

KyungJin Shin

**Table of Contents**

1. Introduction
2. Rules of the Game
3. Summary of your development
4. Example Inputs with Outputs
5. Flowcharts
6. Pseudo-code
7. Reference
8. Completed Check-off Sheet
9. **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

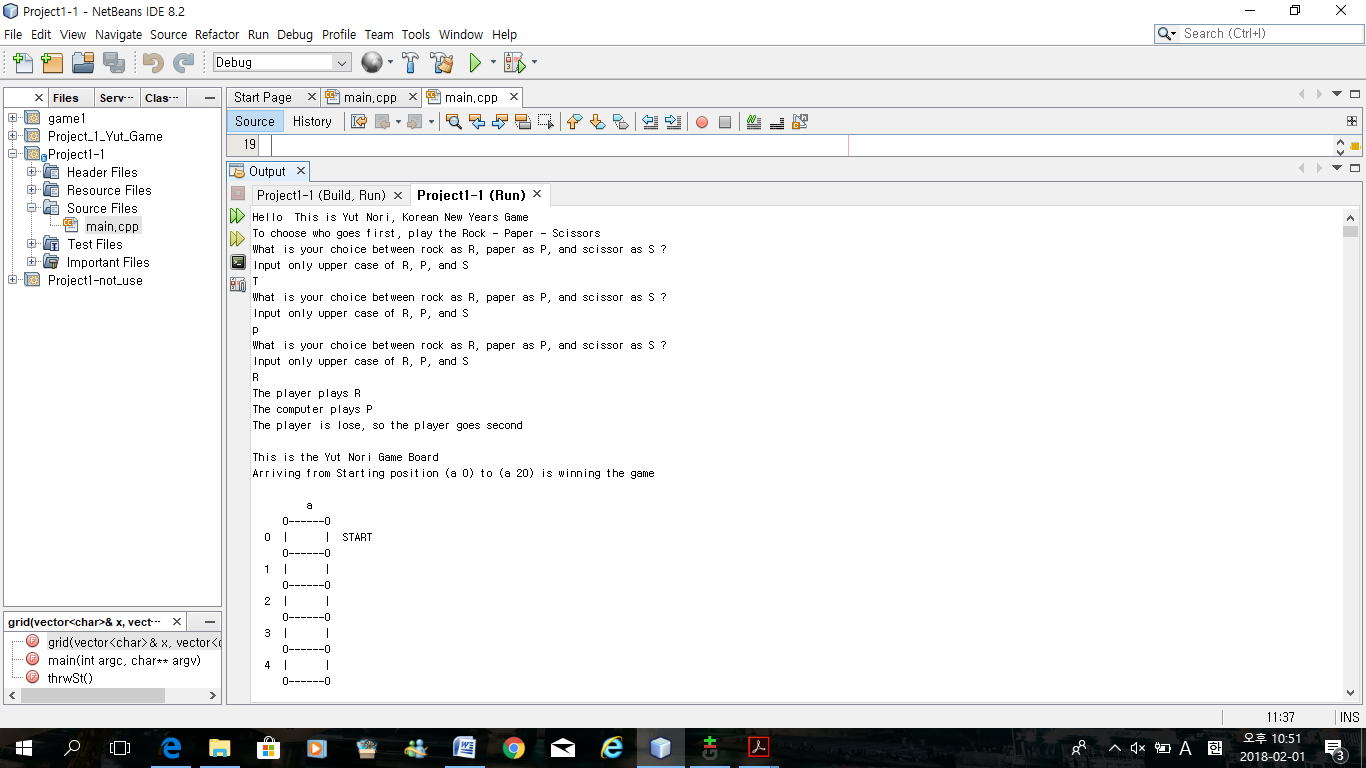
If user and computer are on the same position, second arrived piece could catch the first arrived piece, so second arrived piece will take the place, and the first arrived piece has to go back to the starting point.

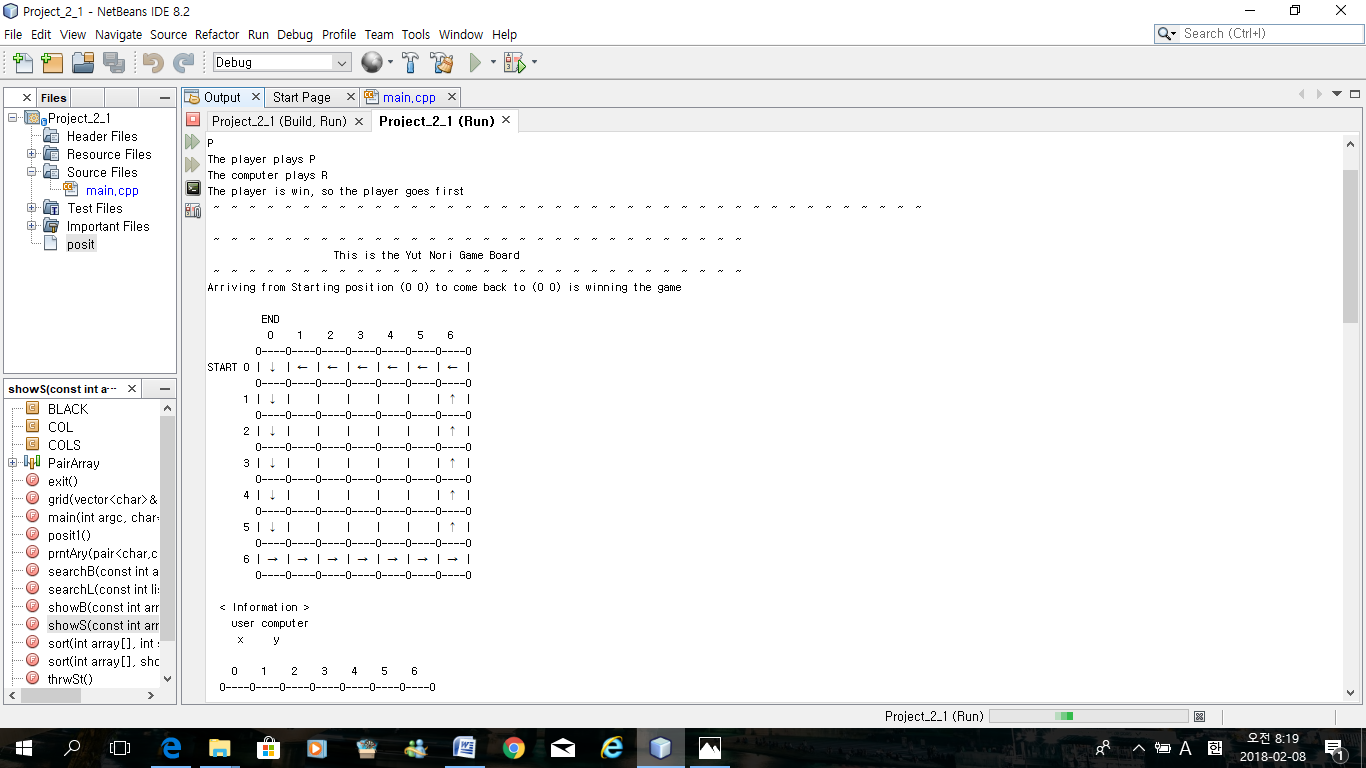
1. **Summary of your development**

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).



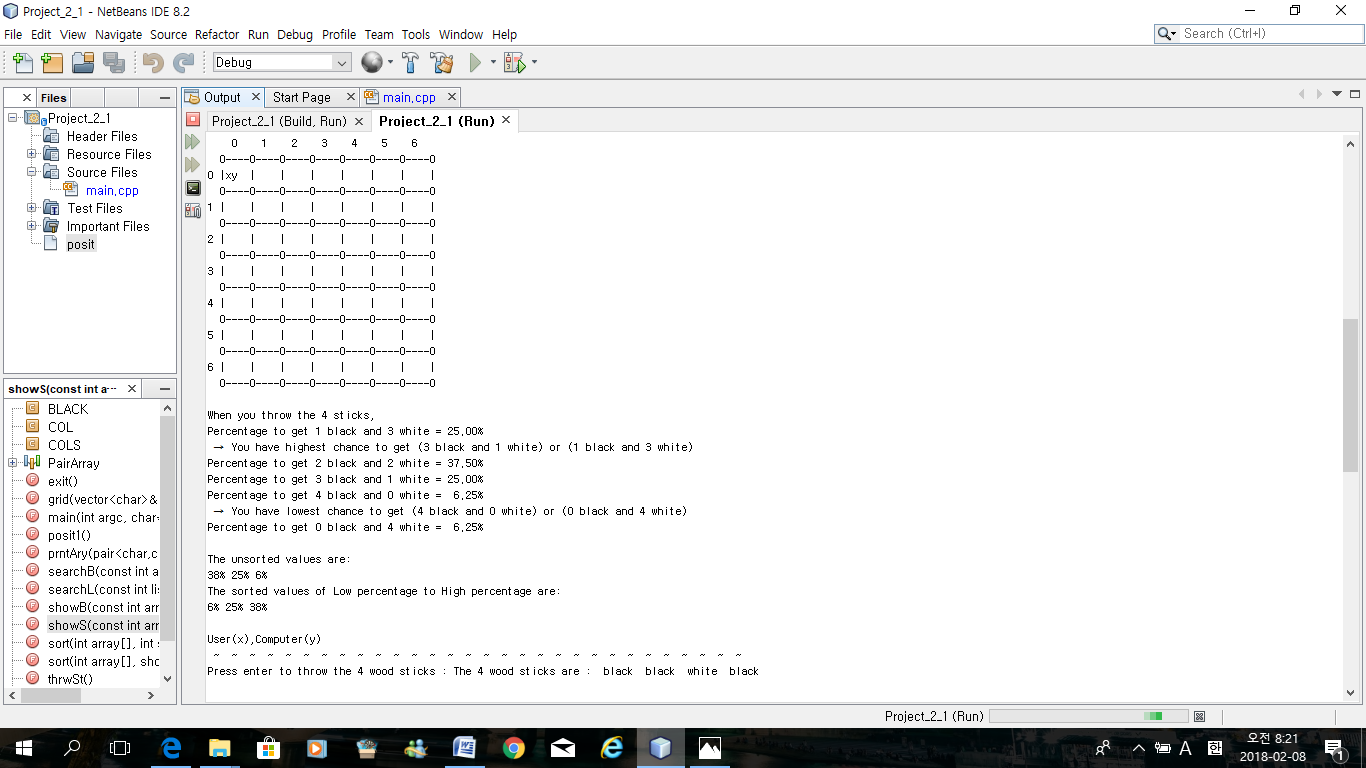


This is the possible percentage when you throw the 4 sticks each time.

Because the player wins the Rock – Scissor – Paper game, Player will go first.

Show the path way grid first.

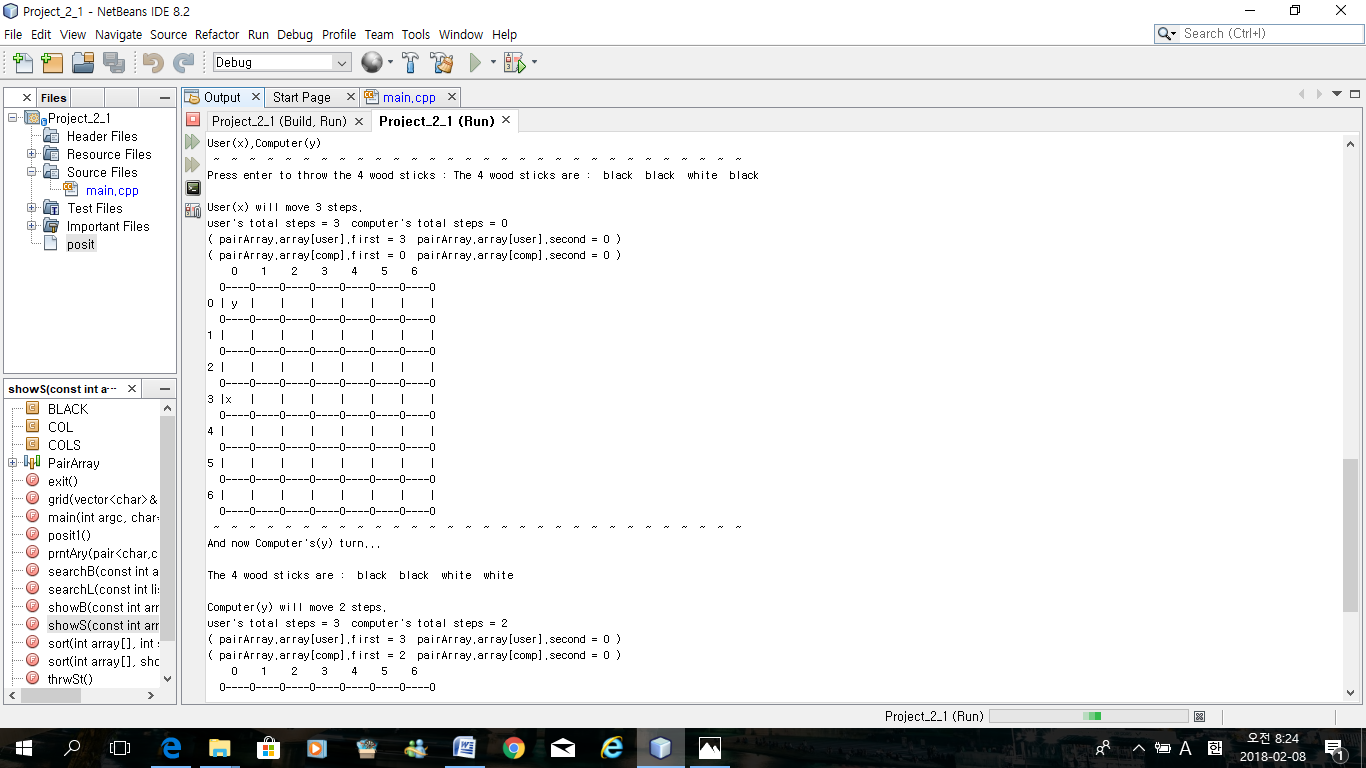
Declare user’s piece is x and computer’s piece is y.



This board shows starting position of x and y.

Also, percentage of getting different number of black and white when user throw the 4 sticks (like a flip a coin).

Throwing the sticks one more time is only around 6% possible.



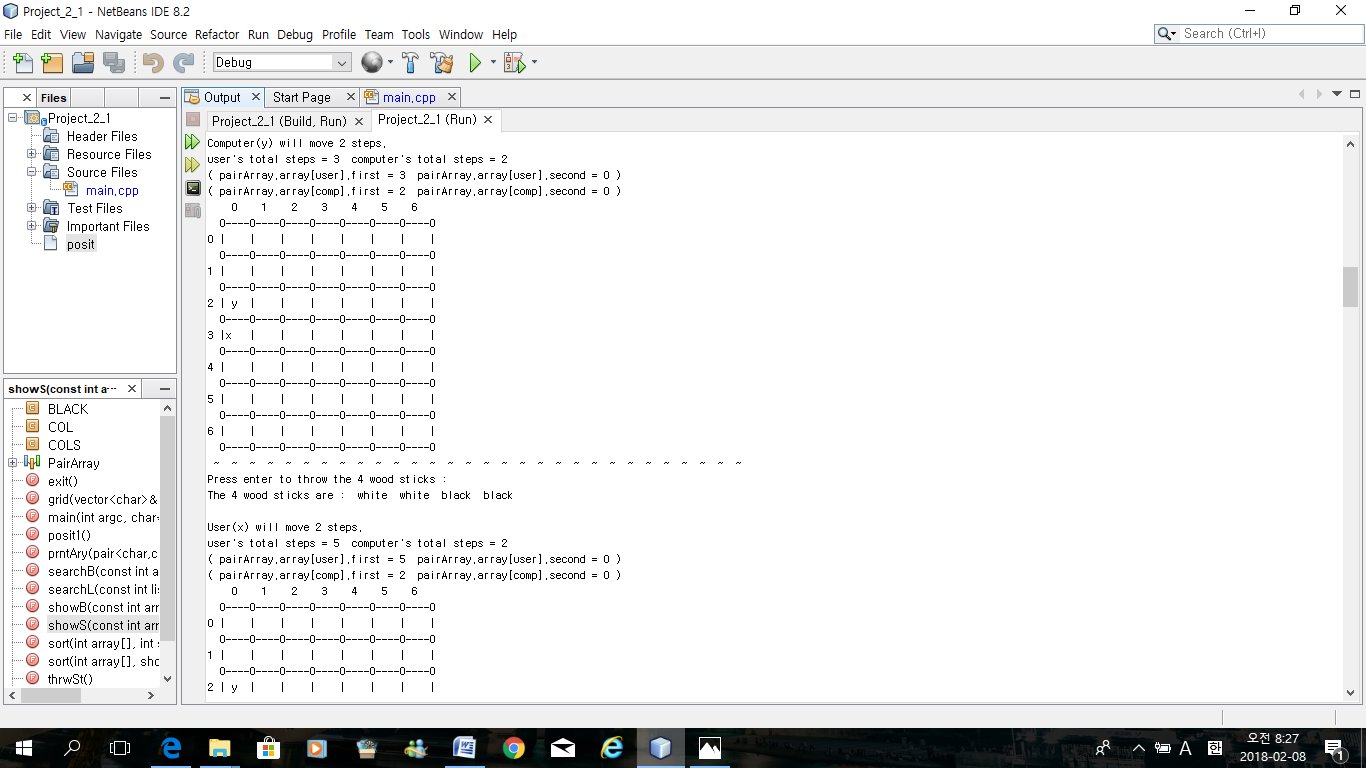
Press Enter key only to throw the 4 sticks.

User got 3 black and 1 white, so user will moves 3 steps.

It displays that new (x) on the board.

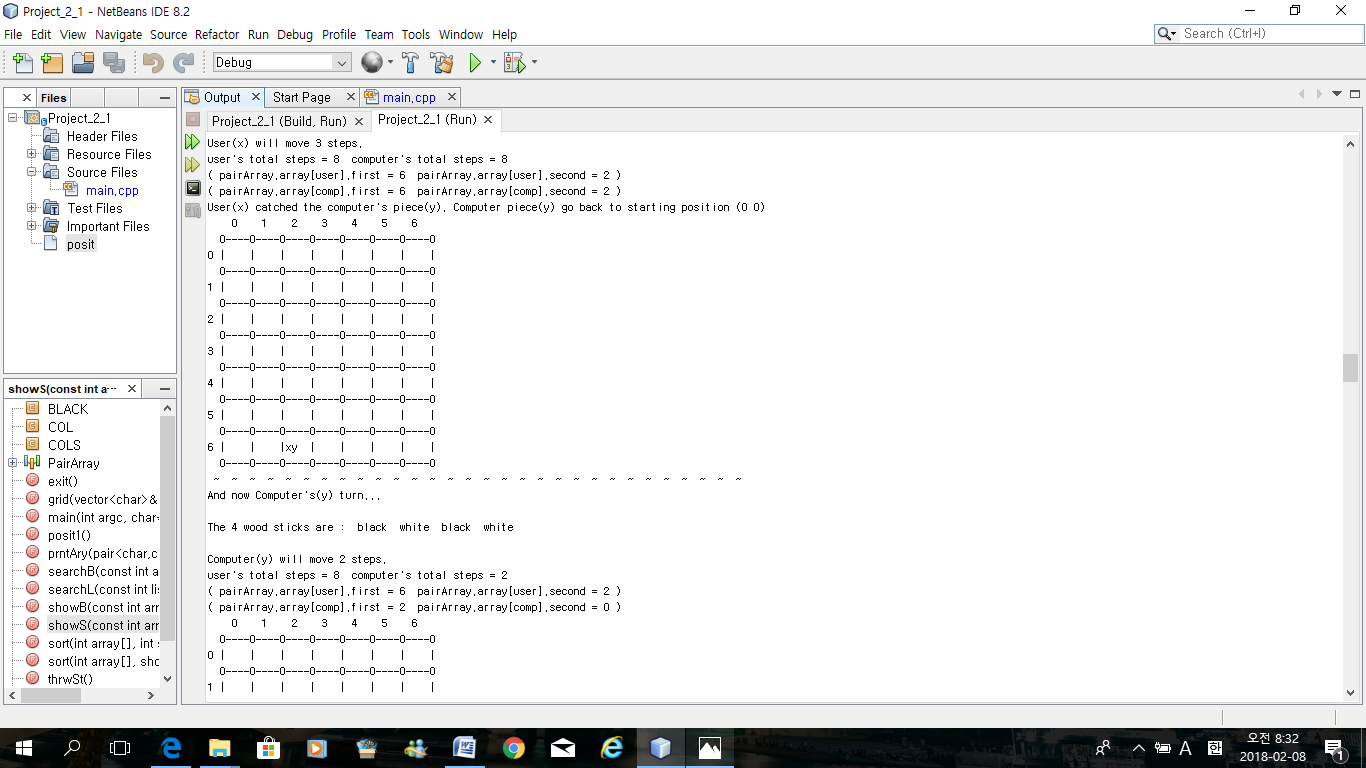
After user’s result, Computer automatically throw the sticks.

Computer will move 2 steps.



Now, on the new board, x on the 3 steps and y on the 2 steps.

Continue pressing Enter key.

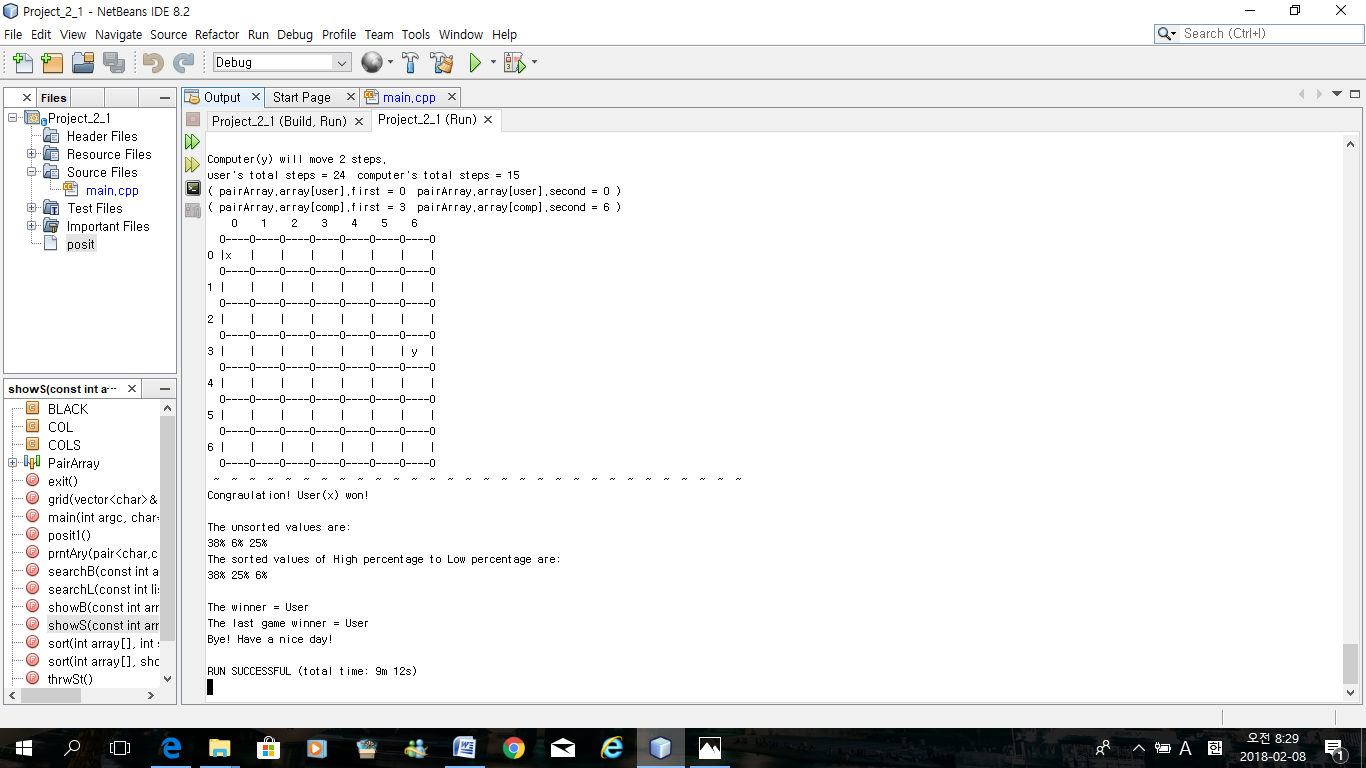


If computer and user are on the same position,

Computer (y) was first, and User(x) catches the y.

User will stay the new position, but computer has to go back to original position (0,0).

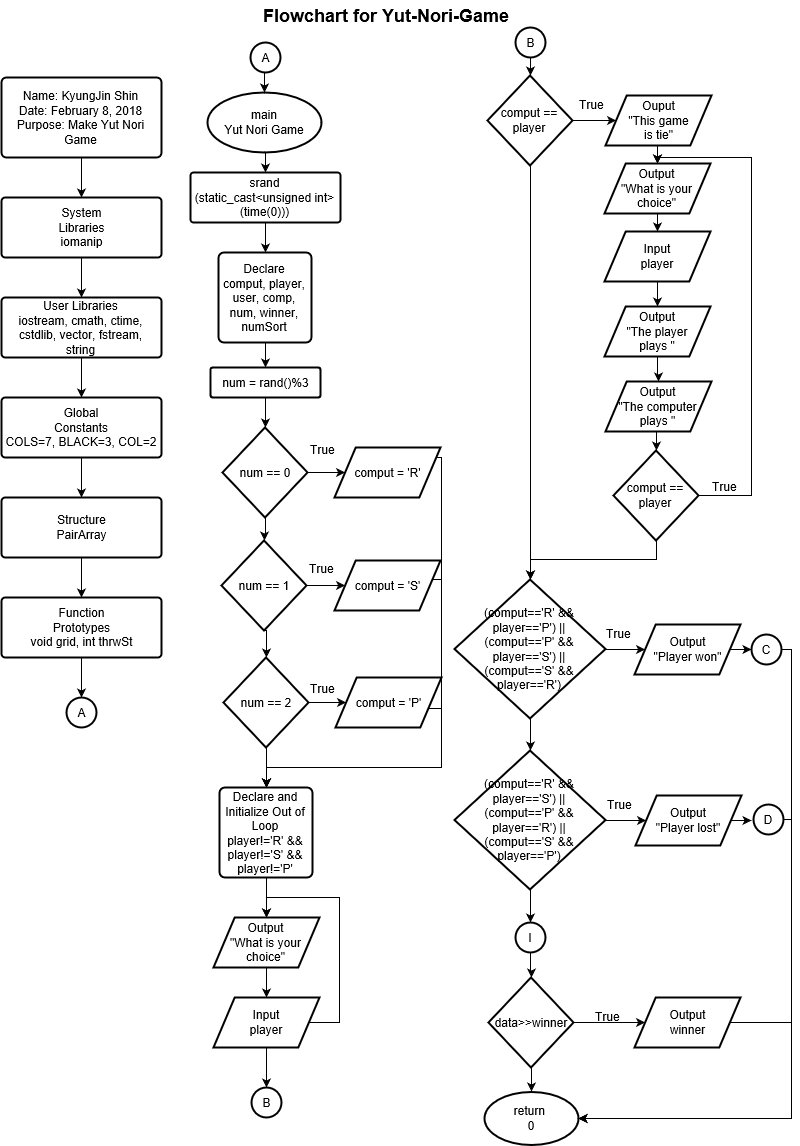
And then, computer will throw the sticks to move on from starting point (0,0)



User (x) arrived / will pass the endpoint (0,0) first, so User won!

38% is the highest chance to move 1 steps or 3 steps when you throw the sticks.

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



1. **Flowcharts**

Because my code is so long, I break the flow chart as ternary operator, independent if, dependent if else, switch, for loop, do-while loop, while loop, and function.

This flow chart is

Opening the comments.

Bring system libraries.

Bring function prototypes.

Starts main.

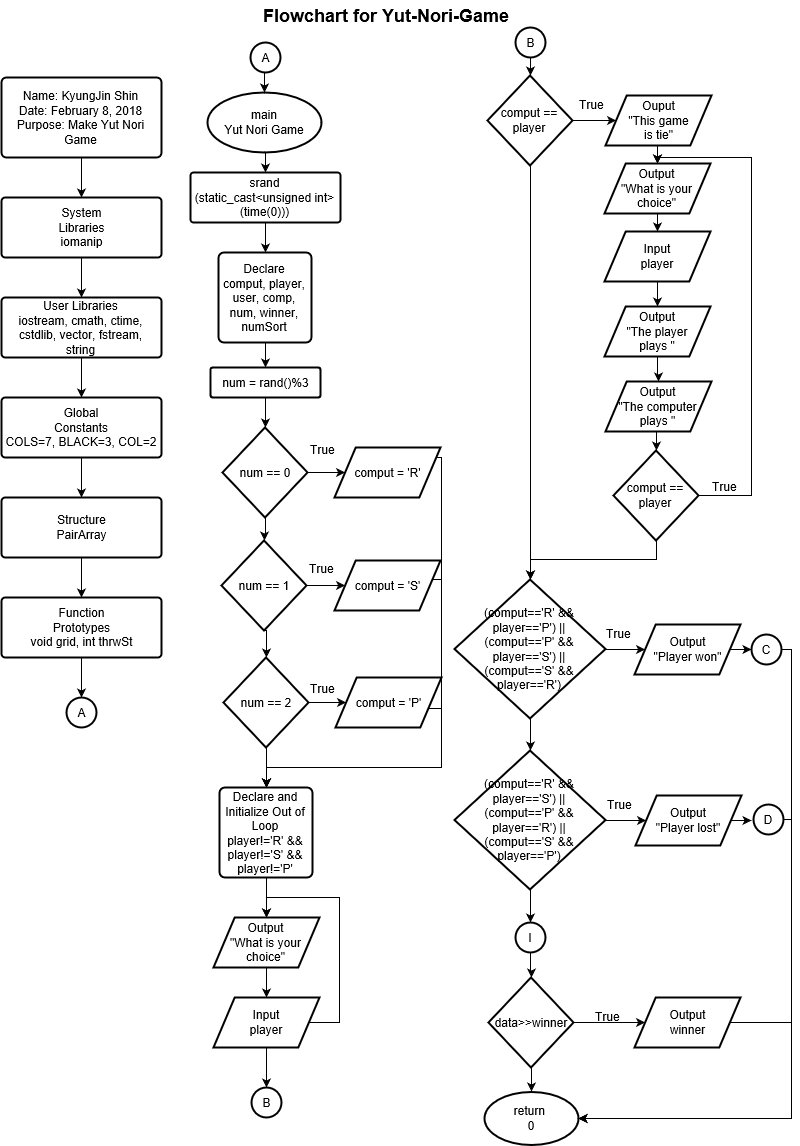
Set the random number seed.

Declare variables.

Switch case (0 to 2)

While loop until user input R, S, or

P with upper case.

Independent If – if computer and player are same

Output the result that the game is tie.

And using Do – While loop to give a chance

to user to input again. Loop this until user’s

input is different than computer’s choice.

Dependent If else – If

player input ‘P’ and computer chose ‘R’,

player input ‘S’ and computer chose ‘P’,

player input ‘R’ and computer chose ‘S’.

Output the Player won -> C

Player goes first, and computer goes next.

Else if

player input ‘R’ and computer chose ‘P’,

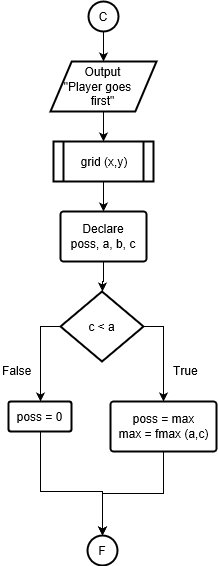
player input ‘P’ and computer chose ‘S’,

player input ‘S’ and computer chose ‘R’.

Output the Player lost -> D

Computer goes fist, and player goes next.

Return to 0.

Player won -> C

Output the player goes first.

Open the grid(x,y) Function.

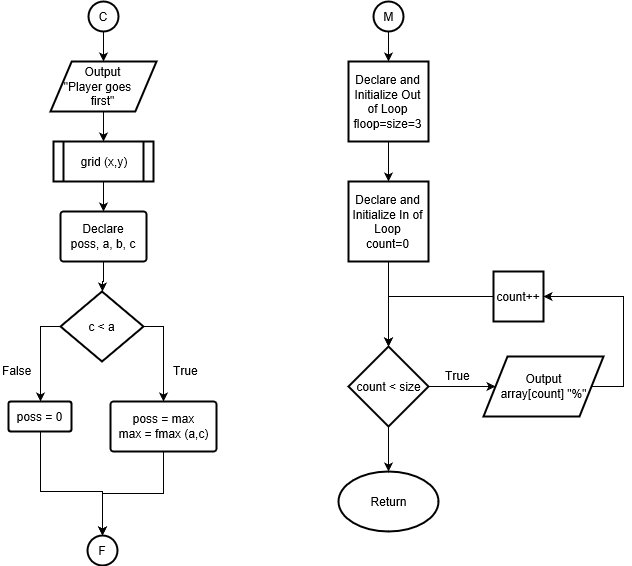
Ternary operator

If ( c ) is less than ( a ) -> true : poss is same as max

And max is same as fmax(a,c).

->false : poss is equal to 0.

F -> Output the percentage of throwing 4 sticks

M -> Function void show

For loop – Declare SIZE = 3

Declare count =0

If count is smaller than SIZE,

* True : Output array[count ] “%”

Increment count one

Continue this loop until

SIZE is bigger than count.

* False : return nothing because

void function

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

//#include <algorithm> //Linear Searching - not necessary

using namespace std;

//User Libraries

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

// 2-D Array Dimensions

const int COLS=7; //Total 7 Columns for print Array and game board

const int BLACK = 3; //Up to 3 black for 2D array of black and white sticks

const int COL=2; //Information(declare x and y), 2-D array

//structure //When I make library to struct PairArray, sometimes run successful and run failed

struct PairArray{ //However, now it runs successfully

pair<int, int> \*array; //position of the array, \* as pointer

int size; //size of the array

};

**//Function Prototypes**

int searchL(const int [], int, int); //Linear searchlist

int searchB(const int [], int, int); //Binary searchlist

bool sort(int [], short); //Bubble sort array

void showB(const int [], int); //Bubble sort show array

void sort(int [], int); //Selection sort

void showS(const int [], int); //Selection show array

void tilde(int = 20, int = 1); //20 (tilde ~~~) col and 1 row

void twoDAry(string [][COL], int); //Show and declare that user=x, and computer=y

void grid(vector<char> &, vector<char> &); //Show the path of board movement

int thrwSt(); //Throw the sticks(dice)

PairArray posit1(); //(0,0) -> 0, (1,0) -> 1 PairArray File

void exit(); //To exit the program

void prntAry(pair<char, char> board[][COLS],int ROWS ); //[][COLS] == 2-D

//Execution Begins Here

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

//Generate Random Number Seed

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

**//Declare Variables**

const int ROWS=7; //Total 7 Rows for print Array and game board

const int ROW=2; //Information(declare) 2-D array

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

short numSort = 3;

//Declare, Initialize position on the board

pair<int, int> player1( 0, 0 ); //Initially player1(user) stars at (0,0)

pair<int, int> player2( 0, 0 ); //Initially player2(computer) stars at (0,0)

//User is x, computer is y by using 2-D array

string info[ROW][COL]={{ " user" , "computer" }, //Show user is x, and computer is y

{ " x" , " y" }};

**//Example grid path way**

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-------------------------------------------------\*/**

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; //If c is less than a -> max, if not -> 0

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

//Linear search

const int SIZE=3; //Because there are only 3 numbers (25,38,6)

int highCh[SIZE] = {25,38,6}; //25%, 37.5% to round up 38%, 6.25% to round down 6%, search list, randomly ordered

int resultL; //Result of Linear search

resultL = searchL(highCh,SIZE,25); //Linear search, find 25 from 25,38,and 6

//Binary search

int lowCh[SIZE] = {6,25,38,}; //25%, 37.5% to round up 38%, 6.25% to round down 6%, search list

int resultB; //Result of Binary search

resultB = searchB(lowCh,SIZE,6);

//2-D array for number of white and number of black side of sticks

const int WHITE = 3; //Up to 3 white

int chance[BLACK][WHITE]; //2D Array

//Display stars to indicate Rock-Scissor-Paper game to Yut Nori Game

tilde(); //print out ~

//Declare board array

pair<char, char> board[ROWS][COLS]; //char because it will print out x and y

//init the board array

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

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

board[i][j] = make\_pair( ' ', ' ' ); //Blank space on each board position

}

}

cout<<endl;

tilde(15); //print out ( ~ tilde) 15 times by using defult argument

cout<<" "<<setw(20)<<" "<<"This is the Yut Nori Game Board"<<endl; //without " ", the title doesn't move to the center

tilde(15); //print out ( ~ tilde) 15 times by using defult argument

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

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

cout<<endl;

twoDAry(info,ROW); const int ROW=2; //Information(declare) 2-D array

cout<<endl; //give a one line space after showing example grid

board[0][0] = make\_pair( 'x', 'y' ); //Starting points, Both x and y start at (row0 col0)

prntAry( board, ROWS ); //Print the string position on the board

//Display percentage of throwing 4 sticks

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

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

for(int black=BLACK; black<=BLACK; black++){ //2-D array for number of white and number of black side of sticks

for(int white=WHITE; white>=WHITE; white--){

cout<<"Percentage to get "<<(black-2)<<" black and "<<(white)

<<" white = "

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

if(resultL == -1){

cout<<" → Your did not get (3 black and 1 white) or

(1 black and 3 white)"<<endl; //Linear search - not found

}else {

cout<<" → You have highest chance to get (3 black and 1 white)

or (1 black and 3 white)"<<endl; //Linear search - found

}

}

}

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

if(resultB == -1){

cout<<" → Your did not get (4 black and 0 white) or (0 black and 4 white)"<<endl; //Binary search - not found

}else {

cout<<" → You have lowest chance to get (4 black and 0 white) or (0 black and 4 white)"<<endl; //Binary search - found

}

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

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

**//Bubble sort**

//Unsorted values

int valueB[3] = {38,25,6}; //not arranged yet

//Display the values

cout<<"The unsorted values are: "<<endl;

showB(valueB,3);

//Sort the values

if(!sort(valueB, numSort)) { //repeat swap is true, false-> return true, repeat until true, //not false//swap inside the function

//Display them again

cout<<"The sorted values of Low percentage to High percentage are: "<<endl;

showB(valueB,3);

}

cout<<endl; //give a one line space after showing example grid

/**/printArray function**

PairArray posit1(); //Show the position on the grid

//posit function to get that posit array

comp=0; //Initialize computer is 0

user=0; //Initialize user is 0

int count=0; //Initialize count is 0

//initialize posit

PairArray pairArray = posit1();

**//Player goes first**

cout<<"User(x),Computer(y)"<<endl;

//User's piece is x, Computer's piece is y on the board grid

do{

tilde(15); //print out ~~~~ to separate the boards, so user can look easily

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

//Other key won't work. only enter key works

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

do {

count = thrwSt(); //Throw stick function

board[pairArray.array[user].first][pairArray.array[user].second]

= make\_pair( ' ', ' ' );

//Remove the previous pieces on the previous position

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

if( user < pairArray.size ) { //This if, without this, i got sometime failed and successful result. not over size 24.

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

cout << "user's total steps = " << user << " computer's total steps = " << comp << endl;

cout << "( pairArray.array[user].first = " << pairArray.array[user].first << " pairArray.array[user].second = " << pairArray.array[user].second << " )" << endl;

cout << "( pairArray.array[comp].first = " << pairArray.array[comp].first << " pairArray.array[comp].second = " << pairArray.array[comp].second << " )" << endl;

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( 'x', ' ' ); //x moves

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', 'y' ); //y stays

if( user == comp ){

//When user position and computer position are same

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( 'x', 'y' ); //show x and y on the same position

cout<<"User(x) catched the computer's piece(y). Computer piece(y) go back to starting position (0 0)"<<endl;

count=0;//Initialize computer's count is 0

comp=0; //Initialize computer is 0. Computer go back to starting position because user's piece catched the computer's piece

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

prntAry( board, ROWS ); //Print the Grid. Show new marks on the grid

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( 'x', ' ' ); //x moves

}else prntAry( board, ROWS ); //Print the Grid

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

} // End of check that user < pairArray.size; if i get steps(user) more than 24, it will stop the game

}while((count==4 || count==5) && user < pairArray.size ); //Loops until sticks are white,white,white,white or black,black,black,black

**//Computer goes next**

tilde(15); //print out ~~~~ to separate the boards, so user can look easily

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

do{

count = thrwSt(); //Throw stick function

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

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

if(comp < pairArray.size ) { //This if, without this, i got sometime failed and successful result. not over size 24.

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

cout << "user's total steps = " << user << " computer's total steps = " << comp << endl;

cout << "( pairArray.array[user].first = " << pairArray.array[user].first << " pairArray.array[user].second = " << pairArray.array[user].second << " )" << endl;

cout << "( pairArray.array[comp].first = " << pairArray.array[comp].first << " pairArray.array[comp].second = " << pairArray.array[comp].second << " )" << endl;

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', 'y' ); //y moves

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( 'x', ' ' ); //x moves

if( comp == user ){ //When user position and computer position are same

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( 'x', 'y' ); //show x and y on the same position

cout<<"Computer(y) catched the user's piece(x). User's piece(x) go back to starting position (0 0)"<<endl;

count=0;//Initialize count is 0

user=0; //Initialize computer is 0. Computer go back to starting position because user's piece catched the computer's piece

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

prntAry( board, ROWS ); //Print the Grid. Show new marks on the grid

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( ' ', 'y' ); //y moves

}else prntAry( board, ROWS ); //Print the Grid

if (count==4 || count==5 ) cout<<"Computer(y) 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

} // End of check that comp < pairArray.size;

}while((count==4 || count==5) && comp < pairArray.size); //Loops until sticks are white,white,white,white or black,black,black,black

if (user>=23){ //When user passed the 0,0 position

tilde(15); // shows ~~~~

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

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

}else if (comp>=23){ //When computer passed the 0,0 position

tilde(15); // shows ~~~~

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

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

}

}while(!(comp>=23) && !(user>=23)); //Loops stop when computer piece or user piece arrived at end of the position

//same as = while(!(comp>=23) || !(user>=23)) = while(comp<23 || user<23)

//Selection sort

//Define an array with unsorted values

//const int SIZE = 3; declare before Lineary search

int valueS[SIZE] = {38,6,25};

//Display the values

cout<<endl<<"The unsorted values are: "<<endl;

showS(valueS,SIZE);

//Sort the values

sort(valueS,SIZE);

//Display the values again

cout<<"The sorted values of High percentage to Low percentage are: "<<endl;

showS(valueS,SIZE);

cout<<endl;

//Delete the structure

delete pairArray.array;

/\*------------------------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-------------------------------------------------\*/

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; //If c is less than a -> max, if not -> 0

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

//Linear search

const int SIZE=3; //Because there are only 3 numbers (25,38,6)

int highCh[SIZE] = {25,38,6}; //25%, 37.5% to round up 38%, 6.25% to round down 6%, search list, randomly ordered

int resultL; //Result of Linear search

resultL = searchL(highCh,SIZE,25); //Linear search, find 25 from 25,38,and 6

//Binary search

int lowCh[SIZE] = {6,25,38,}; //25%, 37.5% to round up 38%, 6.25% to round down 6%, search list

int resultB; //Result of Binary search

resultB = searchB(lowCh,SIZE,6);

//2-D array for number of white and number of black side of sticks

const int WHITE = 3; //Up to 3 white

int chance[BLACK][WHITE]; //2D Array

//Display stars to indicate Rock-Scissor-Paper game to Yut Nori Game

tilde(); //print out ~

//Declare board array

pair<char, char> board[ROWS][COLS]; //char because it will print out x and y

//init the board array

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

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

board[i][j] = make\_pair( ' ', ' ' ); //Blank space on each board position

}

}

cout<<endl;

tilde(15); //print out ( ~ tilde) 15 times by using defult argument

cout<<" "<<setw(20)<<" "<<"This is the Yut Nori Game Board"<<endl; //without " ", the title doesn't move to the center

tilde(15); //print out ( ~ tilde) 15 times by using defult argument

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

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

cout<<endl;

twoDAry(info,ROW); const int ROW=2; //Information(declare) 2-D array

cout<<endl; //give a one line space after showing example grid

board[0][0] = make\_pair( 'x', 'y' ); //Starting points, Both x and y start at (row0 col0)

prntAry( board, ROWS ); //Print the string position on the board

//Display percentage of throwing 4 sticks

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

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

for(int black=BLACK; black<=BLACK; black++){ //2-D array for number of white and number of black side of sticks

for(int white=WHITE; white>=WHITE; white--){

cout<<"Percentage to get "<<(black-2)<<" black and "<<(white)<<" white = "

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

if(resultL == -1){

cout<<" → Your did not get (3 black and 1 white) or (1 black and 3 white)"<<endl; //Linear search - not found

}else {

cout<<" → You have highest chance to get (3 black and 1 white) or (1 black and 3 white)"<<endl; //Linear search - found

}

}

}

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

if(resultB == -1){

cout<<" → Your did not get (4 black and 0 white) or (0 black and 4 white)"<<endl; //Binary search - not found

}else {

cout<<" → You have lowest chance to get (4 black and 0 white) or (0 black and 4 white)"<<endl; //Binary search - found

}

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

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

//Bubble sort

//Unsorted values

int valueB[3] = {38,25,6}; //not arranged yet

//Display the values

cout<<"The unsorted values are: "<<endl;

showB(valueB,3);

//Sort the values

if(!sort(valueB, numSort)) { //repeat swap is true, false-> return true, repeat until true, //not false//swap inside the function

//Display them again

cout<<"The sorted values of Low percentage to High percentage are: "<<endl;

showB(valueB,3);

}

cout<<endl; //give a one line space after showing example grid

//printArray function

PairArray posit1(); //Show the position on the grid

//posit function to get that posit array

comp=0; //Initialize computer is 0

user=0; //Initialize user is 0

int count=0; //Initialize count is 0

//initialize posit

PairArray pairArray = posit1();

//Computer goes Fist

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

do{

tilde(15); //print out ~~~~ to separate the boards, so user can look easily

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

do{

count = thrwSt(); //Throw stick function

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

//somewhere after you display it and before you throw sticks again

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

if(comp < pairArray.size ) {

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

cout << "user's total steps = " << user << " computer's total steps = " << comp << endl;

cout << "( pairArray.array[user].first = " << pairArray.array[user].first << " pairArray.array[user].second = " << pairArray.array[user].second << " )" << endl;

cout << "( pairArray.array[comp].first = " << pairArray.array[comp].first << " pairArray.array[comp].second = " << pairArray.array[comp].second << " )" << endl;

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', 'y' ); //move y

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( 'x', ' ' ); //move x

if( comp == user ){

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( 'x', 'y' ); //show x and y on the same position

cout<<"Computer(y) catched the user's piece(x). User's piece(x) go back to starting position (0 0)"<<endl;

count=0;//Initialize count is 0

user=0; //Initialize computer is 0. Computer go back to starting position because user's piece catched the computer's piece

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

prntAry( board, ROWS ); //Print the Grid. Show new marks on the grid

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( ' ', 'y' ); //move y

}else prntAry( board, ROWS ); //Print the Computer's Grid

if (count==4 || count==5 ) cout<<"Computer(y) 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

} // End of check that comp < pairArray.size;

}while((count==4 || count==5) && comp < pairArray.size); //Loops until sticks are white,white,white,white or black,black,black,black

//Player goes next

tilde(15); //print out ~~~~ to separate the boards, so user can look easily

cout<<"Press enter to throw the 4 wood sticks : "; //Other key won't work. only enter key works

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

do {

count = thrwSt(); //Throw stick function

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

//board[count][0] = make\_pair( ' ', 'y' ); //make x[position]=' '; somewhere after you display it and before you throw sticks again

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

if( user < pairArray.size ) { //This if, without this, i got sometime failed and successful result. not over size 24.

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

cout << "user's total steps = " << user << " computer's total steps = " << comp << endl;

cout << "( pairArray.array[user].first = " << pairArray.array[user].first << " pairArray.array[user].second = " << pairArray.array[user].second << " )" << endl;

cout << "( pairArray.array[comp].first = " << pairArray.array[comp].first << " pairArray.array[comp].second = " << pairArray.array[comp].second << " )" << endl;

board[pairArray.array[user].first][pairArray.array[user].second] = make\_pair( 'x', ' ' ); //move x

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', 'y' ); //move y

if( user == comp ){

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( 'x', 'y' ); //show x and y on the same position

cout<<"User(x) catched the computer's piece(y). Computer piece(y) go back to starting position (0 0)"<<endl;

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

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( ' ', ' ' ); //Previous mark will disappear

prntAry( board, ROWS ); //Print the Grid //Show new marks on the grid

board[pairArray.array[comp].first][pairArray.array[comp].second] = make\_pair( 'x', ' ' ); //move x

}else prntAry( board, ROWS ); //Print the User's Grid

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

} // End of check that user < pairArray.size; if i get steps(user) more than 24, it will stop the game

}while((count==4 || count==5) && user < pairArray.size ); //Loops until sticks are white,white,white,white or black,black,black,black

if (user>=23){ //When user passed the a20 position

tilde(15); // shows ~~~~

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

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

}else if (comp>=23){ //When computer passed the a20 position

tilde(15); // shows ~~~~

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

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

}

}while(!(comp>=23) && !(user>=23)); //Loops stop when computer piece or user piece arrived at end of the position

//same as = while(!(comp>=23) || !(user>=23)) = while(comp<23 || user<23)

//Selection sort

//Define an array with unsorted values

//const int SIZE = 3; declare before Lineary search

int valueS[SIZE] = {38,6,25};

//Display the values

cout<<endl<<"The unsorted values are: "<<endl;

showS(valueS,SIZE);

//Sort the values

sort(valueS,SIZE);

//Display the values again

cout<<"The sorted values of High percentage to Low percentage are: "<<endl;

showS(valueS,SIZE);

cout<<endl;

//Delete the structure

delete pairArray.array;

}

//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();

exit(); //Bye

return 0;

}

void prntAry( pair<char, char> board[][COLS],int ROWS ){ //[][COLS] == 2-D

//printArray function

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

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

cout << " 0----0----0----0----0----0----0----0"<< endl;

cout << i << " |";

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

cout << board[i][j].first << board[i][j].second <<" "<< '|'; //x,y position

}

cout<<endl;

}

cout << " 0----0----0----0----0----0----0----0"<< endl;

}

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

}

PairArray posit1(){

//Position1 Function

//make the struct

PairArray temp;

//open the file

ifstream file; //Input file

file.open( "posit" ); //Open the file posit inside "Important Files" "posit"

file >> temp.size; //The size is 24, totoal steps(position)

temp.array = new pair<int, int>[temp.size];// posit[positSize];

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

int x, y; //Declare x and y as integer

file >> x;

file >> y;

temp.array[i] = make\_pair( x, y ); //Array will display x and y

}

return temp;

}

void grid(vector<char> &x, vector<char> &y) { //Pass by reference

cout<<setw(14)<<"END "<<endl; //Indicate where is END

cout<<setw(13)<<" 0 1 2 3 4 5 6"<<endl;

cout<<" "<<setw(18)<<right<<" 0----0----0----0----0----0----0----0"<<endl;

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

cout<<setw(3)<<"START"<<setw(2)<<i<<" | "<<"↓"<<" |"<<x[i]<<"←"<<" |"<<x[i]<<"←"<<" |"

<<x[i]<<"←"<<" |"<<x[i]<<"←"<<" |"<<x[i]<<"←"<<" |"<<x[i]<<"←"<<" |"<<endl;

cout<<" "<<setw(18)<<" 0----0----0----0----0----0----0----0"<<endl;

}

for(short i=1; i<6; i++) { // Display positions

cout<<setw(7)<<i<<" | "<<"↓"<<y[i]<<"|"<<x[i]<<" "<<y[i]<<" |"<<x[i]<<" "<<y[i]<<" |"

<<x[i]<<" "<<y[i]<<" |"<<x[i]<<" "<<y[i]<<" |"<<x[i]<<" "<<y[i]<<" |"<<x[i]<<"↑"<<" |"<<endl;

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

cout<<" "<<setw(18)<<" 0----0----0----0----0----0----0----0"<<endl;

}

for(short i=6; i<7; i++) { // Display position

cout<<setw(7)<<i<<" | "<<"→"<<y[i]<<"|"<<" →"<<y[i]<<"|"<<" →"<<y[i]<<"|"

<<" →"<<y[i]<<"|"<<" →"<<y[i]<<"|"<<" →"<<y[i]<<"|"<<" →"<<y[i]<<"|"<<endl;

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

cout<<" "<<setw(18)<<" 0----0----0----0----0----0----0----0"<<endl;

}

}

void twoDAry(string info[][COL], int rows){ //Display and declare user=x, computer=y.

//Heading

cout<<setw(2)<<" "<<"< Information >"<<endl;

for (int x=0; x<rows; x++){

for (int y=0; y<COL; y++){

cout<<setw(4)<<info[x][y]<<" "; //2 x 2

}

cout<<endl;

}

}

void showS(const int array[], int size){ //Show the selection sorting

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

cout<<array[count]<<"% "; //Unit is percentage

}cout<<endl;

}

void sort(int array[], int size){ //Selection sort maximum -> minimum

int scan; //start scan

int maxind; //max index

int maxval; //max value

for(scan = 0; scan < (size - 1); scan++){ //Do not chancge

maxind = scan;

maxval = array[scan];

for(int index = scan+1; index < size; index++){

if(array[index] > maxval){ //if it is min, change to < min :array[index] > maxval

maxval = array[index];

maxind = index;

}

}

array[maxind] = array[scan];

array[scan] = maxval;

}

}

void showB(const int array[], int size){ //Show the Bubble sort

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

cout<<array[count]<<"% "; //Unit is percentage

}cout<<endl;

}

bool sort(int array[], short size){ //Bubble sort

bool swap; //Boolean is only true or false

int temp;

do {

swap=false;

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

if(array[count] > array[count + 1]){

temp = array[count];

array[count] = array[count + 1];

array[count +1 ] = temp;

swap = true;

}

}

}while(swap); //While - true

return swap; //Returning boolean - false

}

int searchB(const int array[], int num, int value){

int first = 0, //First array element

last = num-1, //Last array element

middle, //Midpoint of search

posit = -1; //Position of search value

bool found = false; //Flag

while(!found && first <=last){

middle = (first + last) / 2; //Calculate midpoint

if (array[middle] == value){ //If value is found at mid

found = true;

posit = middle;

}else if (array[middle] > value){ //If value is in lower half

last = middle - 1;

}else { //If value is in upper half

first = middle + 1;

}

}

return posit; //Return to position

}

int searchL(const int list[], int num, int value){

int index = 0; //Used as a subscript to search array

int posit = -1;//Recording Position of search value

bool found = false; //Flag to indicate if the value was found

while(index<num && !found){

if(list[index] == value) { //if the value is found

found = true; //set the flag

posit = index; //position = index, record the value's subscript

}

index++; //go to the next element

}

return posit; //Return the position, or -1

}

void tilde(int cols, int rows){ //Display ~~~ to separate each board and games

for (int down=0; down<rows; down++){

for (int across=0; across<cols; across++){

cout<<" ~ ~ "; //tilde(1) = ~ ~

}cout<<endl;

}

}

void exit(){ //Exit the program

cout<<"Bye! Have a nice day!"<<endl;

exit(0);

}

1. **Reference**
2. Gaddis 8thEd Textbook
3. Dr.Lehr’s Lecture and Lab
4. Michael (Lab hour)
5. **Completed Check-off Sheet**

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