Project 1

<Hide And Seek-Expanded>

CSC-17A 48292
Name: Triet Huynh
Date: 10/29/2022

TABLE OF CONTENTS

Introduction

Title: Hide and Seek

The project is a simple hide-and-seek game that is available in single-player and multiplayer modes.

Upon starting, the player(s) is introduced to a board of nine slots and will have to guess which one is the correct slot that the machine is hiding.

Summary

Project Size: 414 lines.

The number of variables: about 15

The project is a variable of the same game made from CSC 5. The game play and mechanics are initially the same. However, newer concepts and materials from CSC 17A (pointers, C-strings, structures and binary files) were added.

How the game proceeds:

-Originally with single-player mode, the player will have three rounds and as many attempts as it takes to find the hiding spot each round. After each attempt, the game will display the game board with the slot updated as “0” if guessed correctly and “X” otherwise. The program will store the attempts each round in a file named after the player as a record, and it will calculate and announce the average attempts it took per round to the player.

-The project was expanded as we introduced a multiplayer mode and let the players compete to see who wins with fewer attempts on average to locate the hiding slot. A ranking system was also added to show the players’ ranks based on their average attempts.

CSC/CIS 17A Project 1 Check-Off Sheet

Chapter	Section	Concept	Points for Inclusion	Location in Code	Comments
9		**Pointers/Memory Allocation**			
	1	Memory Addresses			
	2	Pointer Variables	5	294	
	3	Arrays/Pointers	5		
	4	Pointer Arithmetic			
	5	Pointer Initialization			
	6	Comparing			
	7	Function Parameters	5	376	
	8	Memory Allocation	5	321	
	9	Return Parameters	5	336	
	10	Smart Pointers			
10		**Char Arrays and Strings**			
	1	Testing			
	2	Case Conversion			
	3	C-Strings	10	116	
	4	Library Functions			
	5	Conversion			
	6	Your own functions			
	7	Strings	10	51	
11		**Structured Data**			
	1	Abstract Data Types			
	2	Data			
	3	Access			
	4	Initialize			
	5	Arrays	5	291	
	6	Nested	5	Stats.h header file	
	7	Function Arguments	5	373	
	8	Function Return	5	336	
	9	Pointers	5	294	
	10	Unions ****			
	11	Enumeration	5	273	
12		**Binary Files**			
	1	File Operations			
	2	Formatting	2	257	
	3	Function Parameters	2	392	
	4	Error Testing			
	5	Member Functions	2	393	
	6	Multiple Files	2	112,289	
	7	Binary Files	5	290	
	8	Records with Structures	5	302	
	9	Random Access Files	5	405	
	10	Input/Output Simultaneous	2	291	
		Total	100		

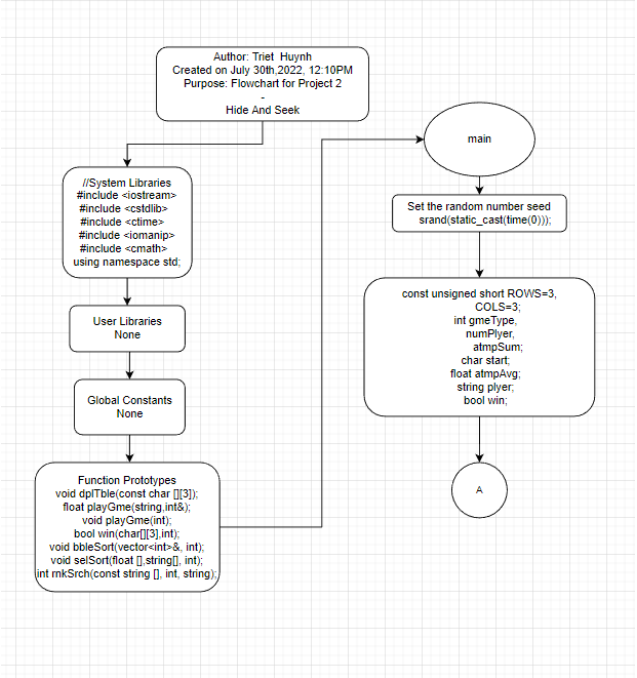
Pseudocodes & Flowcharts

-Opening Comments (name, author, date, purpose)

-Libraries, namespace std

-Function Prototypes

-Main Function
Set random number seed
Declare Variables needed for main



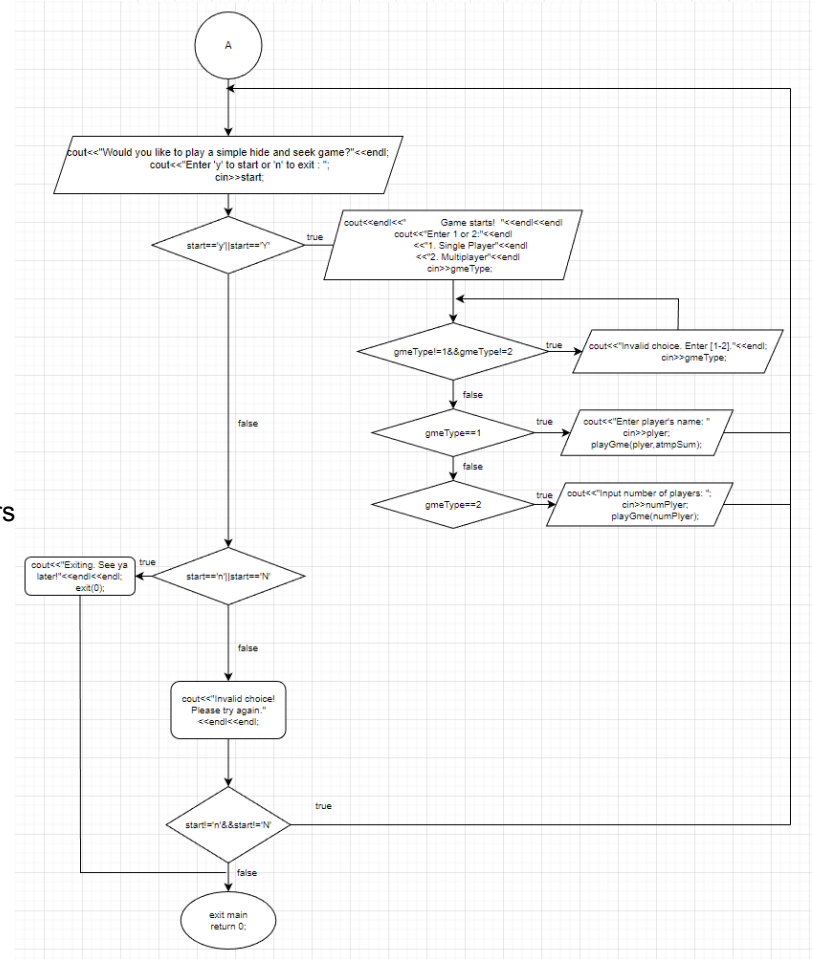
Display games start message and
Get user input into ‘start’

If start is ‘y’ or ‘Y’
Prompt user for single or multiplayer
Verify input with while loop
If single player,
Prompt user for their name &
call single-player game function
If multi-player,
Prompt user for number of players
Call multi-player game function

Else if start is ‘n’ or ‘N’
Display goodbye message and exit

Else
Invalid input

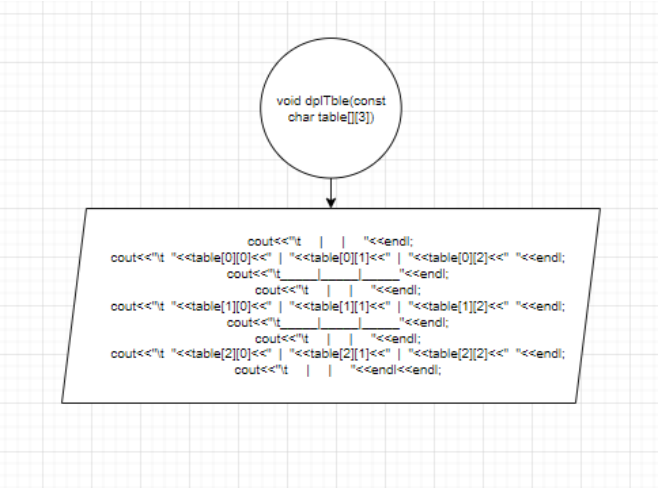
Repeat this in a do-while loop until
Player inputs ‘n’ or ‘N’



-Display table function

Pass in 2D table array and

Output the elements in rows and cols



-Single player game function

Declare and initialize variables needed

Output game rule

Open file to write data to

Use for loop to run 3 game rounds:

Generate a random number

1-9 as hiding slot

Set attempts to 0

Display game table with slots

1-9

Initialize a do-while loop

Prompt user for their guess input

Validate input

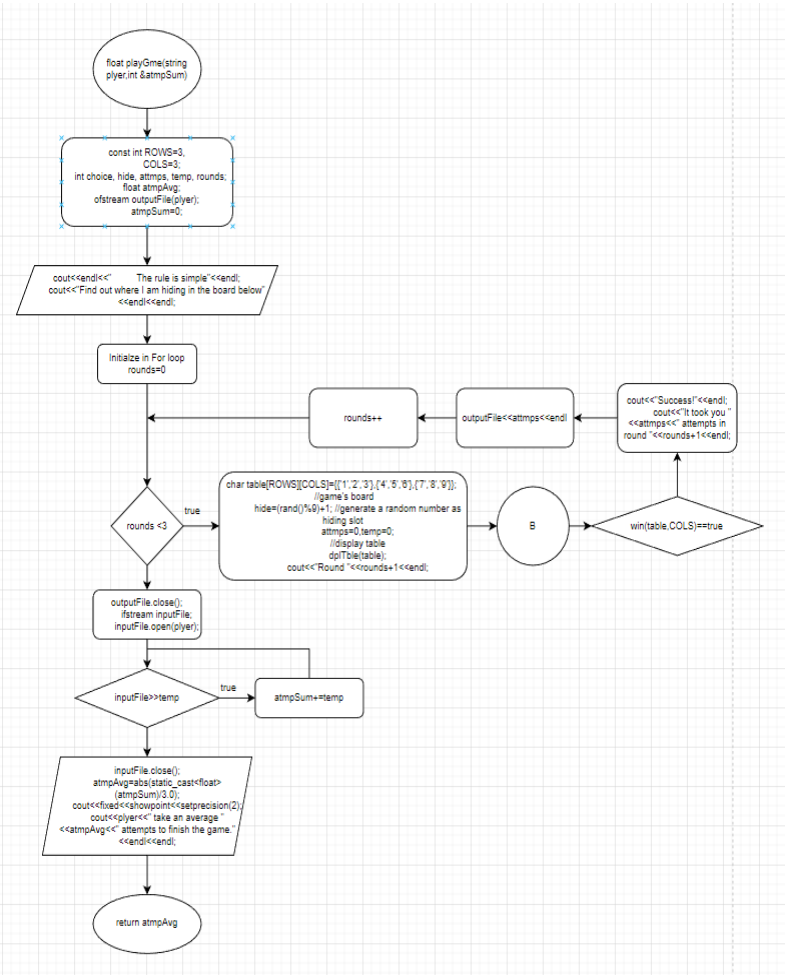
If input<1 and input>9

Display error and prompt user to try again

If input is in range 1-9,

Increment attempt

If input does not match hide,



Replace slot with ‘X’
Display updated table

If input matches hide,
Replace slot with ‘0’
Display updated table
Display success message
Round ends

Loop these steps until user guesses
Correctly and move onto next round

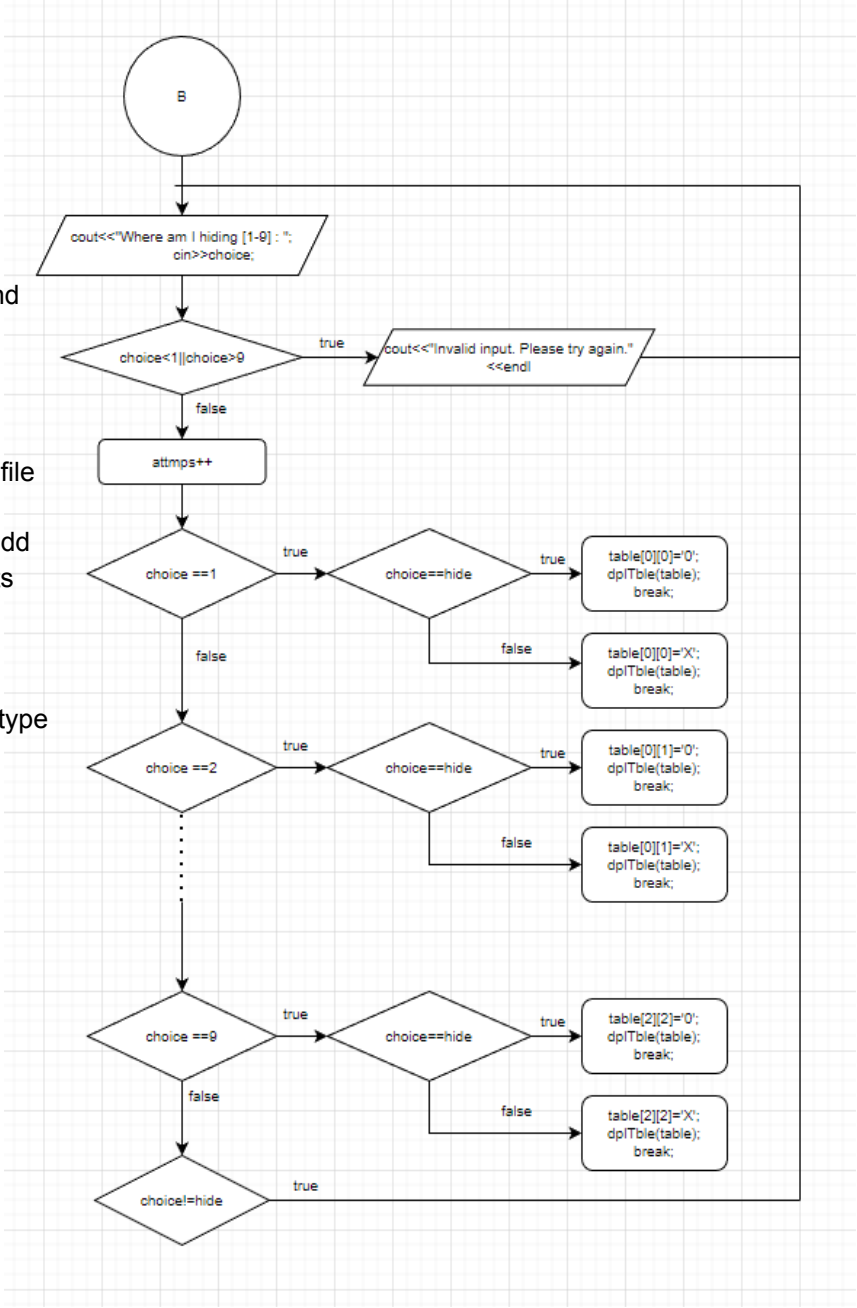
Output number of attempts onto
File after each round

Close output data file and open input data file

Input attempts of 3 rounds from file, then add
And calculate the average attempts

Display average attempts

Return average attempts for float function type

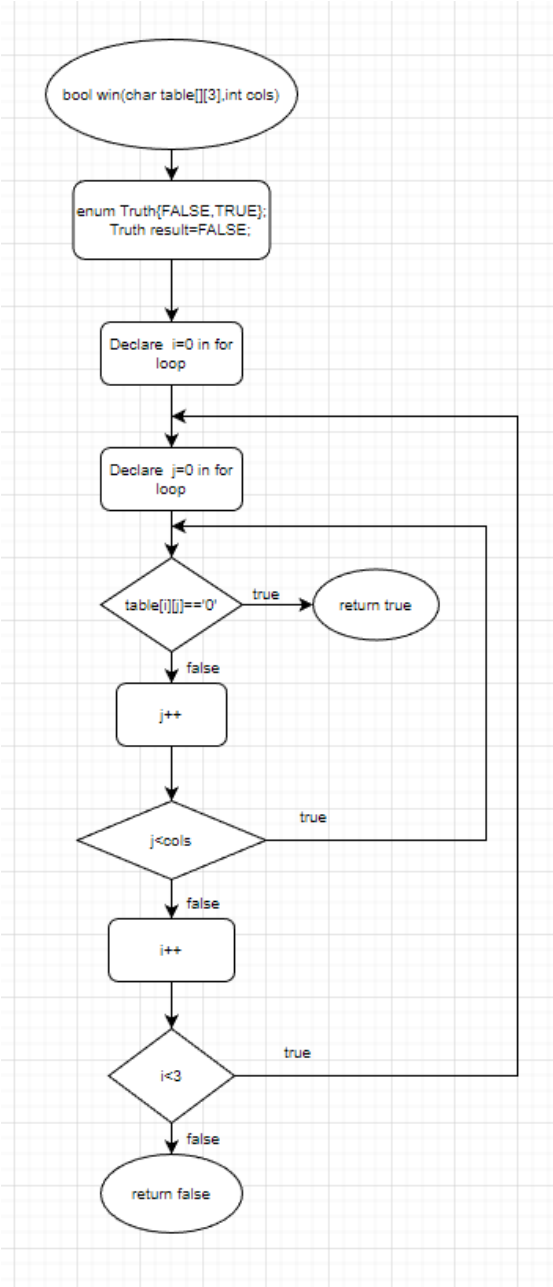


-Win function

Check each of array table's elements

If any of the elements matches '0',
return true

Otherwise return false



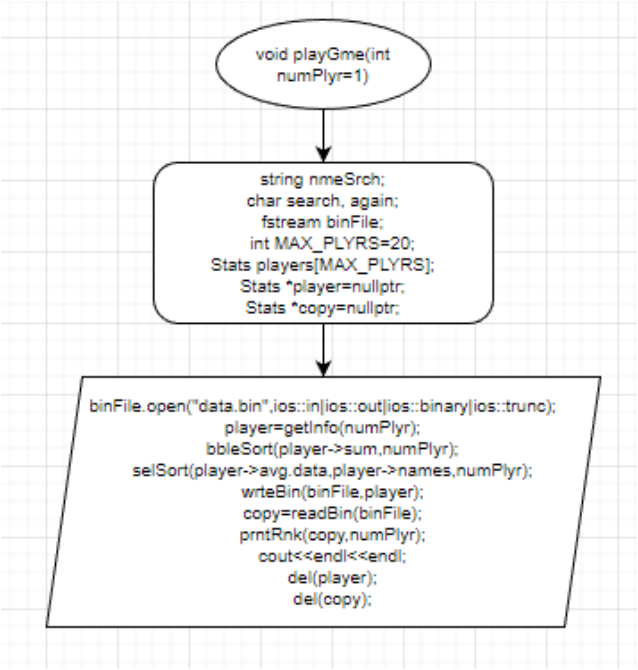
-Multiplayer function

Declare necessary variables & arrays

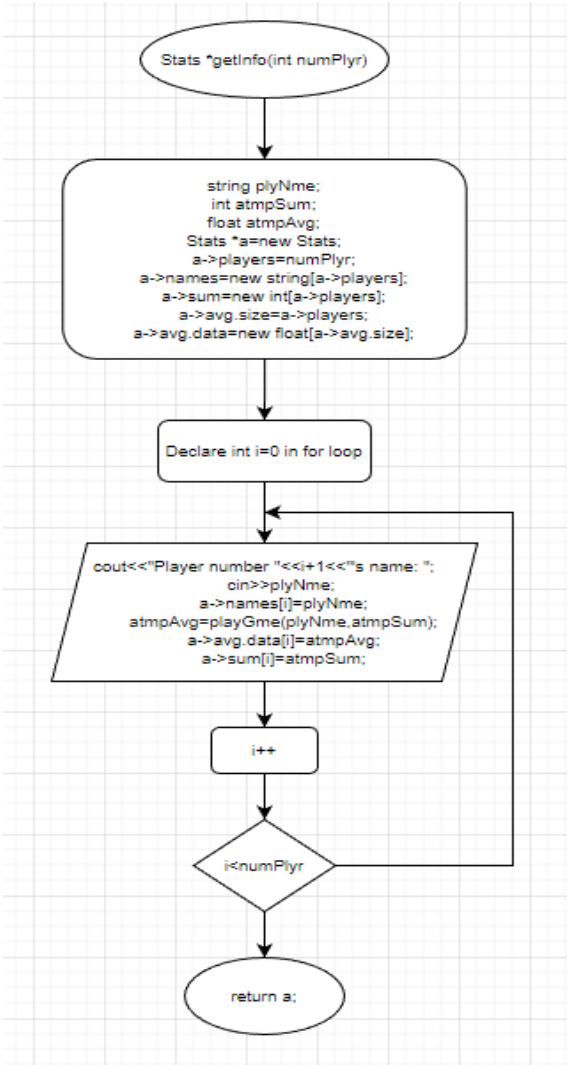
Repeat the single-player game for each
of the players while inputting
their names, total scores and
average scores in arrays and vector

Sort the arrays using selection sort
and vector using bubble sort
in descending order
while making sure they are parallel

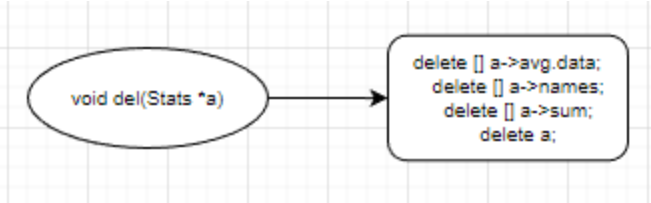
Display the ranking table



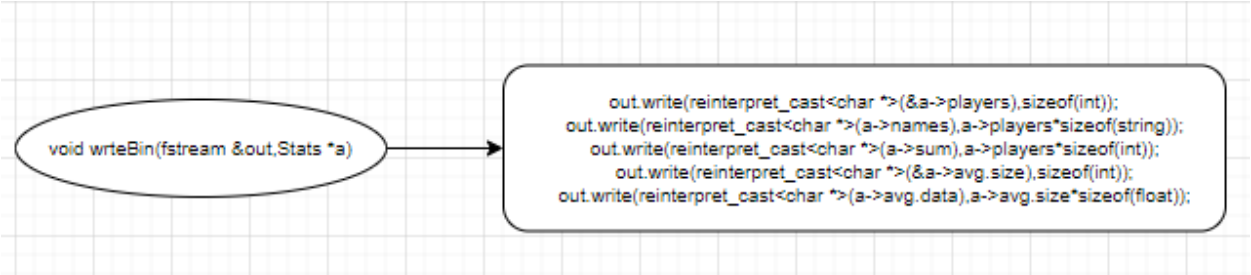
-Get info function
Declare necessary variables,
And pointer to structure
For from first to last player
Get player name
Run single-player function
Store their names, total attempts,
and average number of attemps
in structure
End For
Return pointer to structure



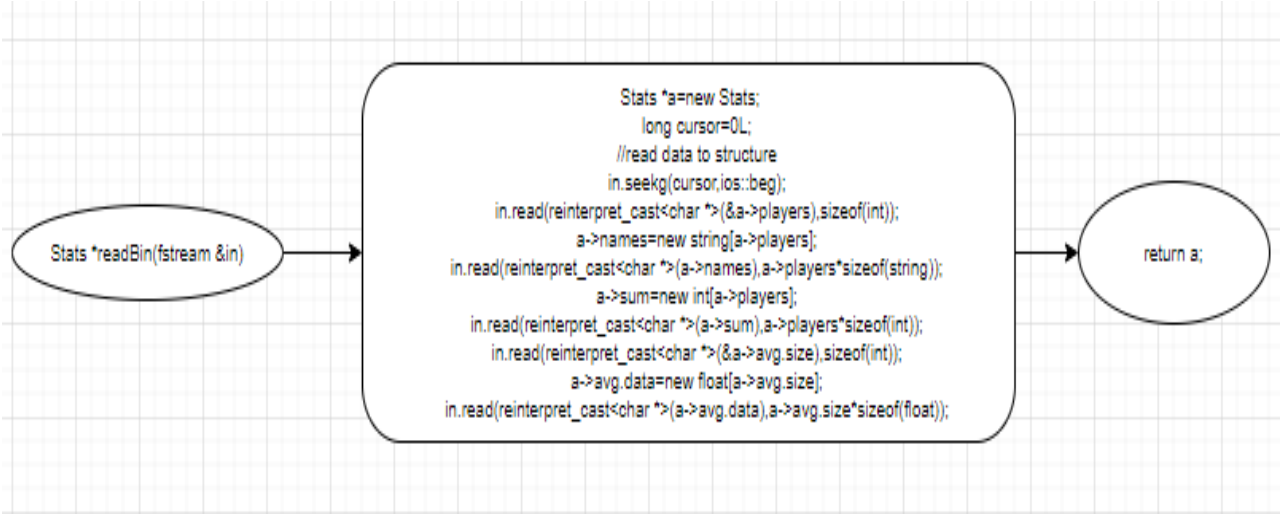
-Delete dynamic allocated structure
pointer function
Delete structure's array components
Delete inner structure's array components
Delete structure



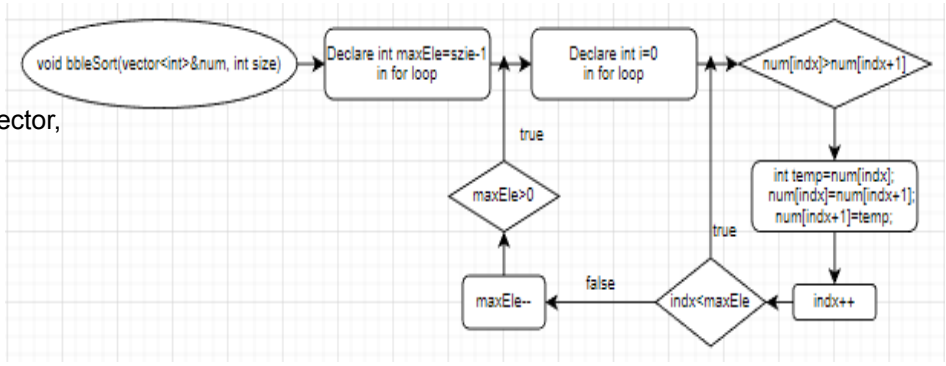
-Write to binary file function
Write each structure components until finished



-Read from binary file function
Set cursor to the beginning of file
Read each structure component
till end of file



-Bubble Sort Function
For maxEle=each subscript in the vector,
from the last to the first
    For index=0 to maxEle-1
      If vector[index]>vector[index+1]
         Swap vector[index] with
                vector[index+1]
      End if
    End for
End for



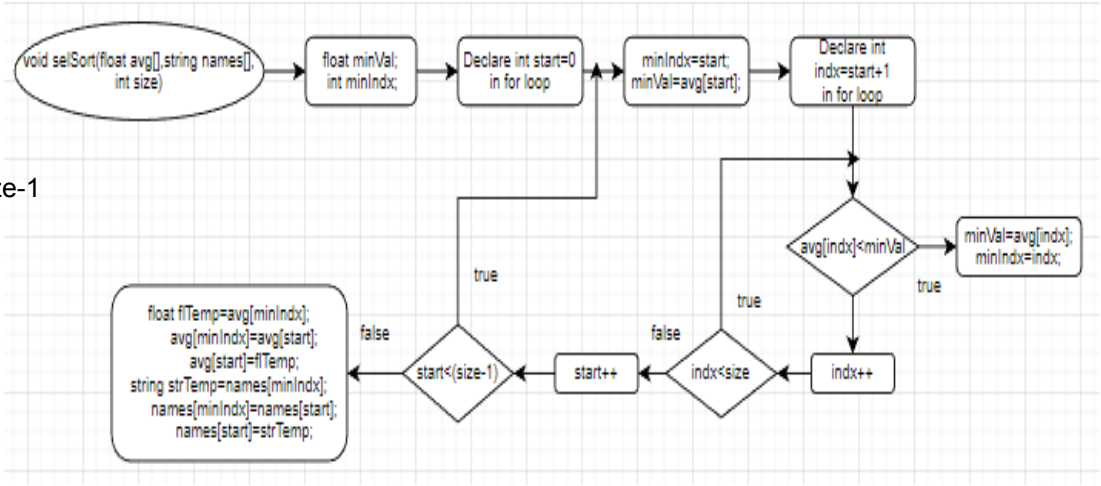
-Selection Sort Function
For start=each array avg subscript,
from the first to last-1

    minIndx=start
    minVal=avg[start]

   For index=start+1 to size-1

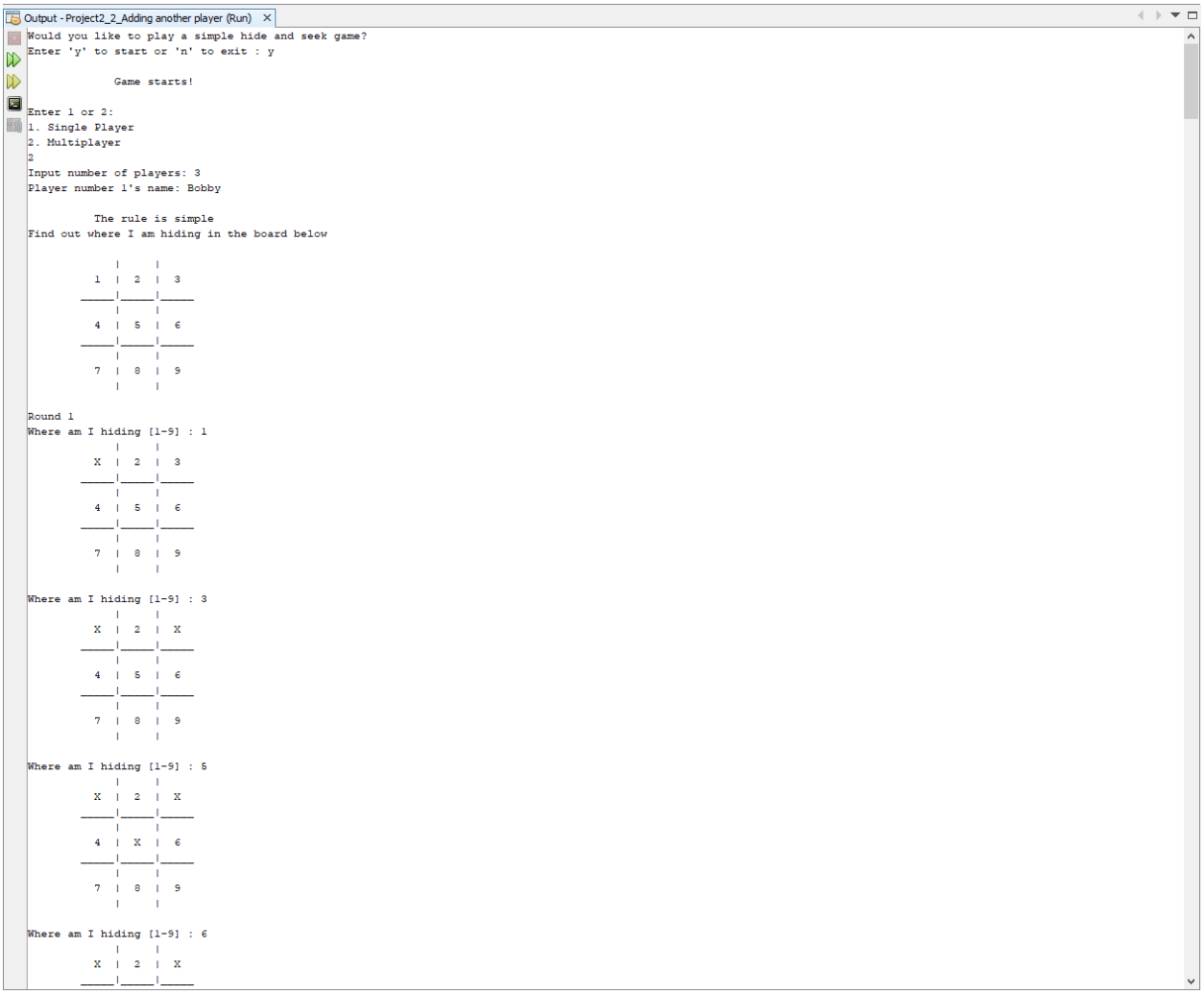
     If avg[indx]<minVal
        minVal=avg[indx]
        minIndx=indx

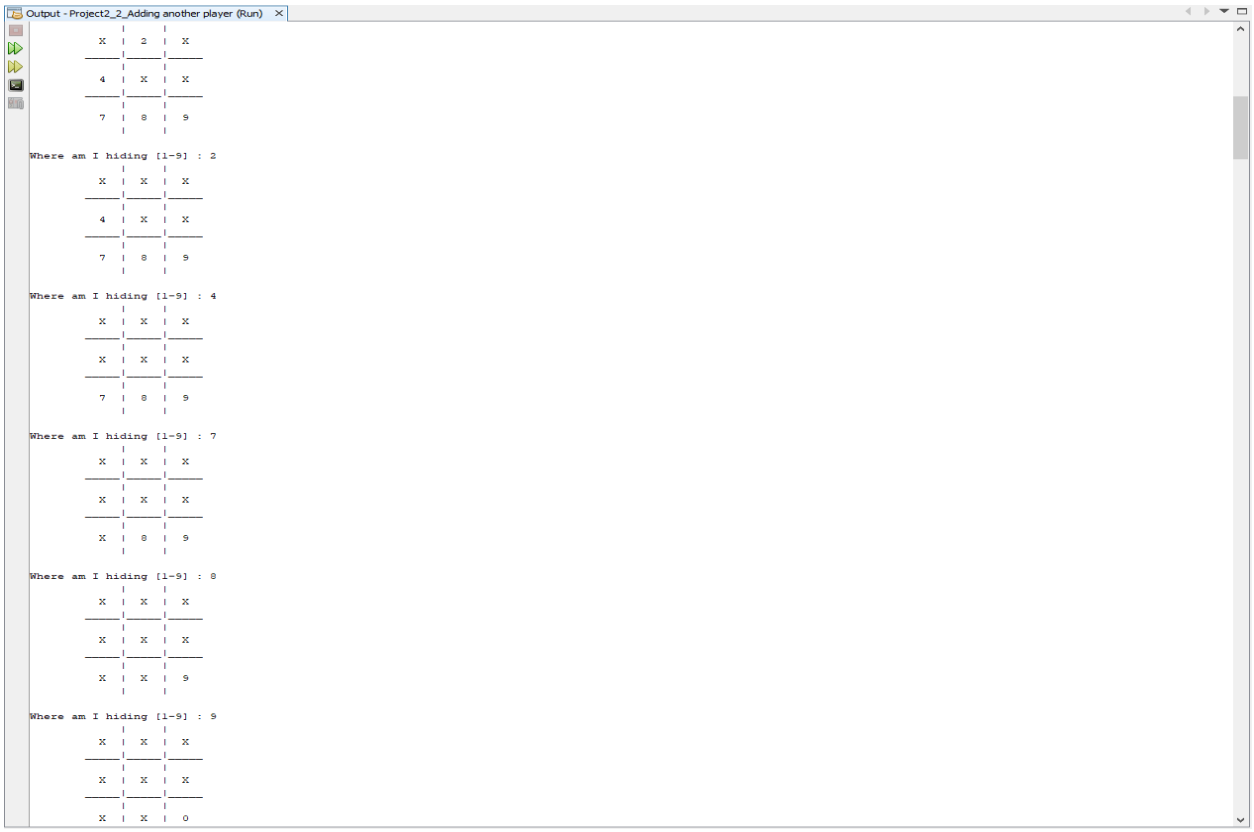
     End if
    End For

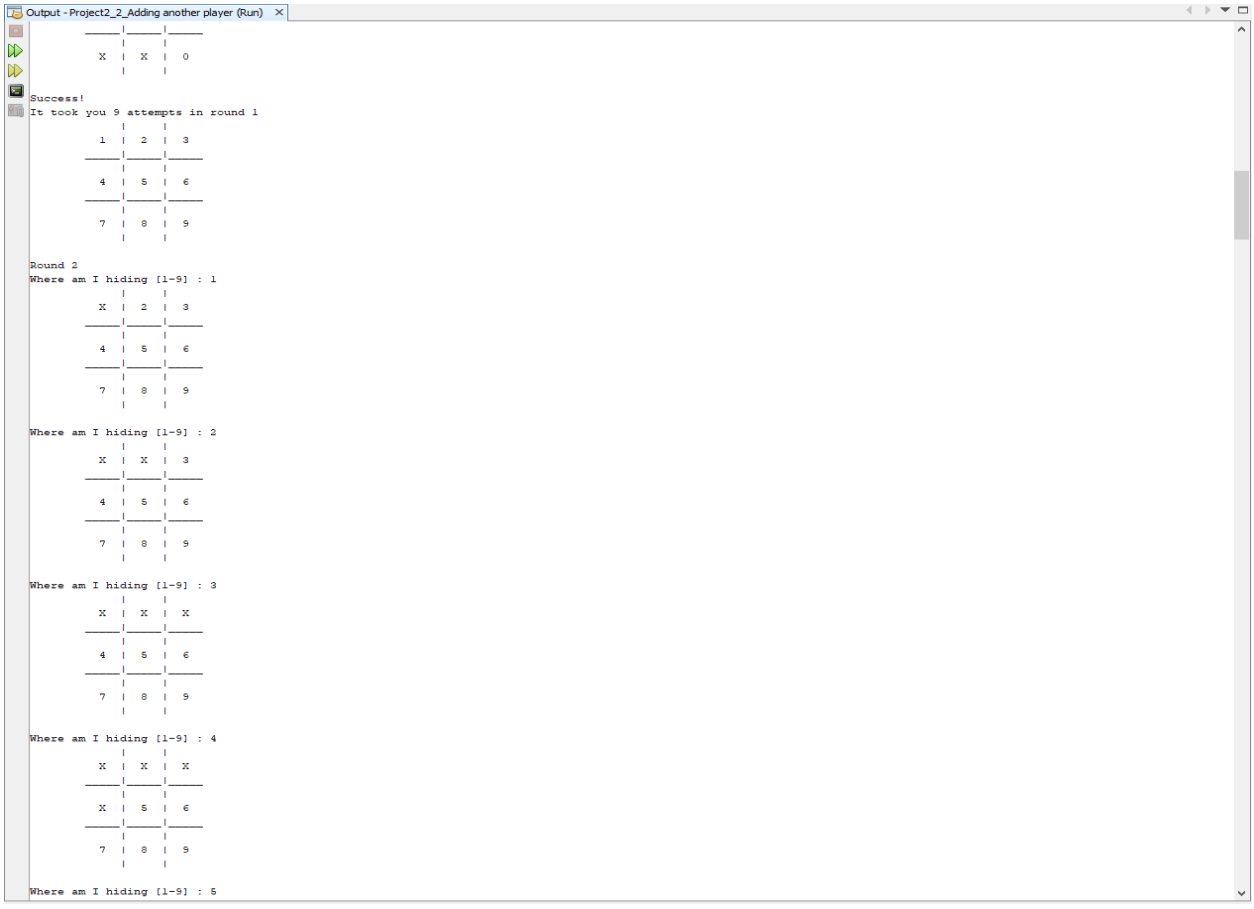


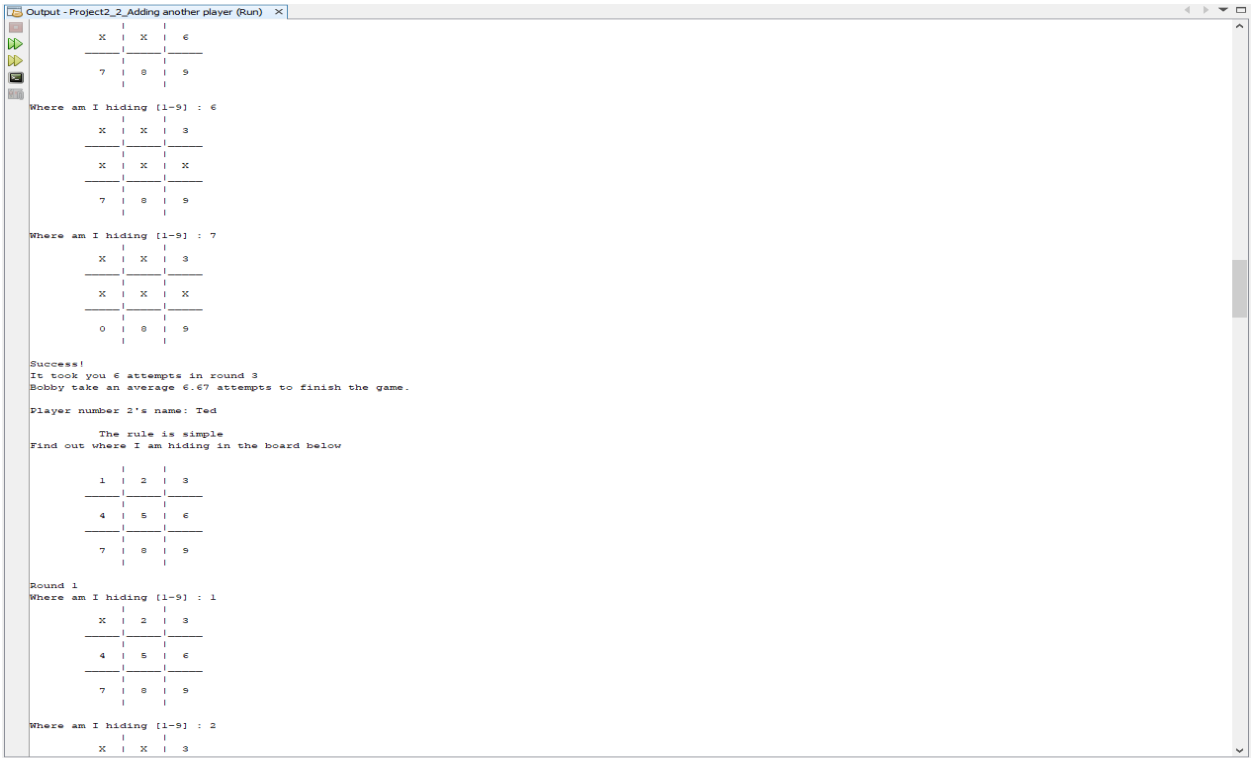
    Swap avg[minIndx] with avg[start]
    Swap names[minIndx] with names[start]
End For

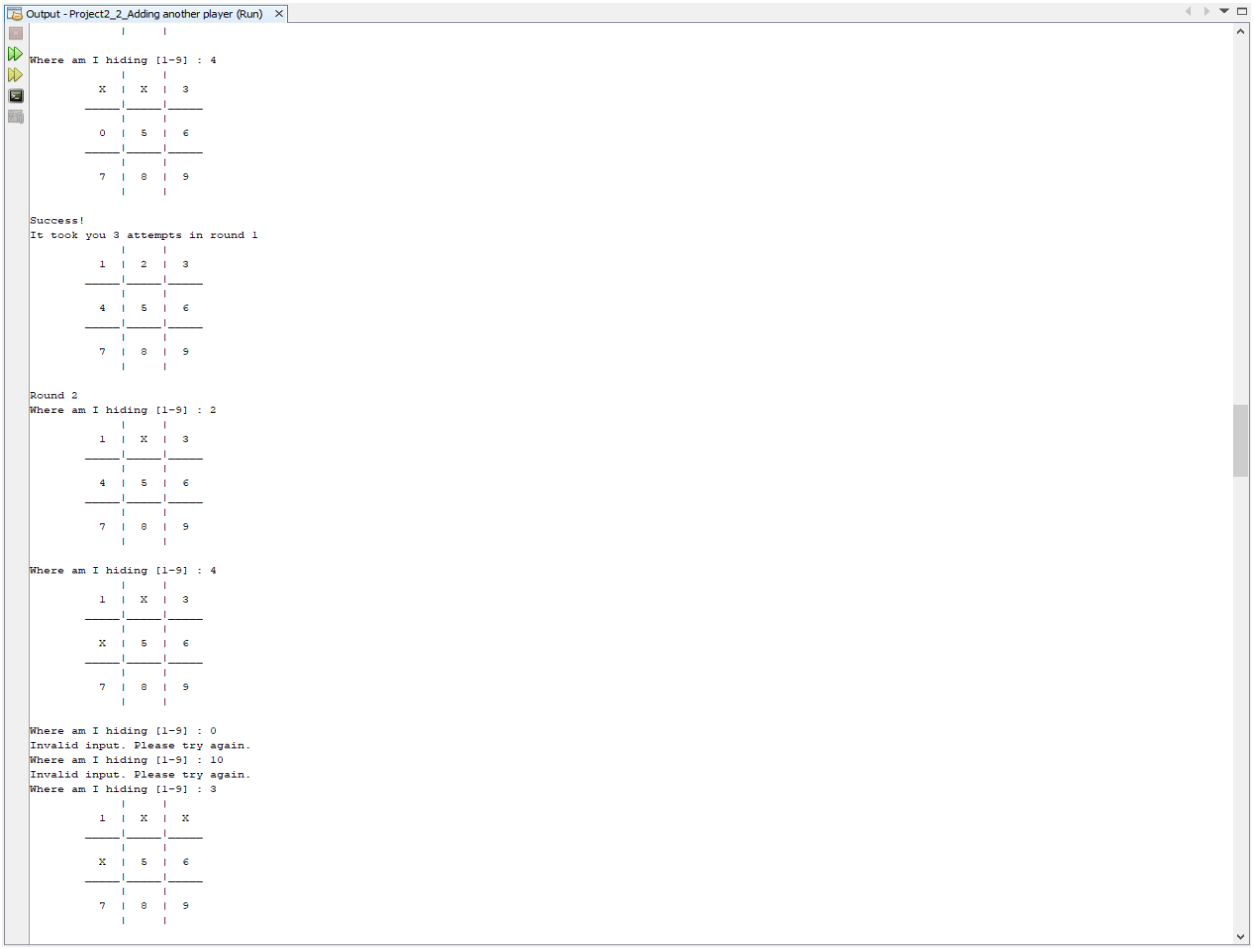
Proof of a working Program

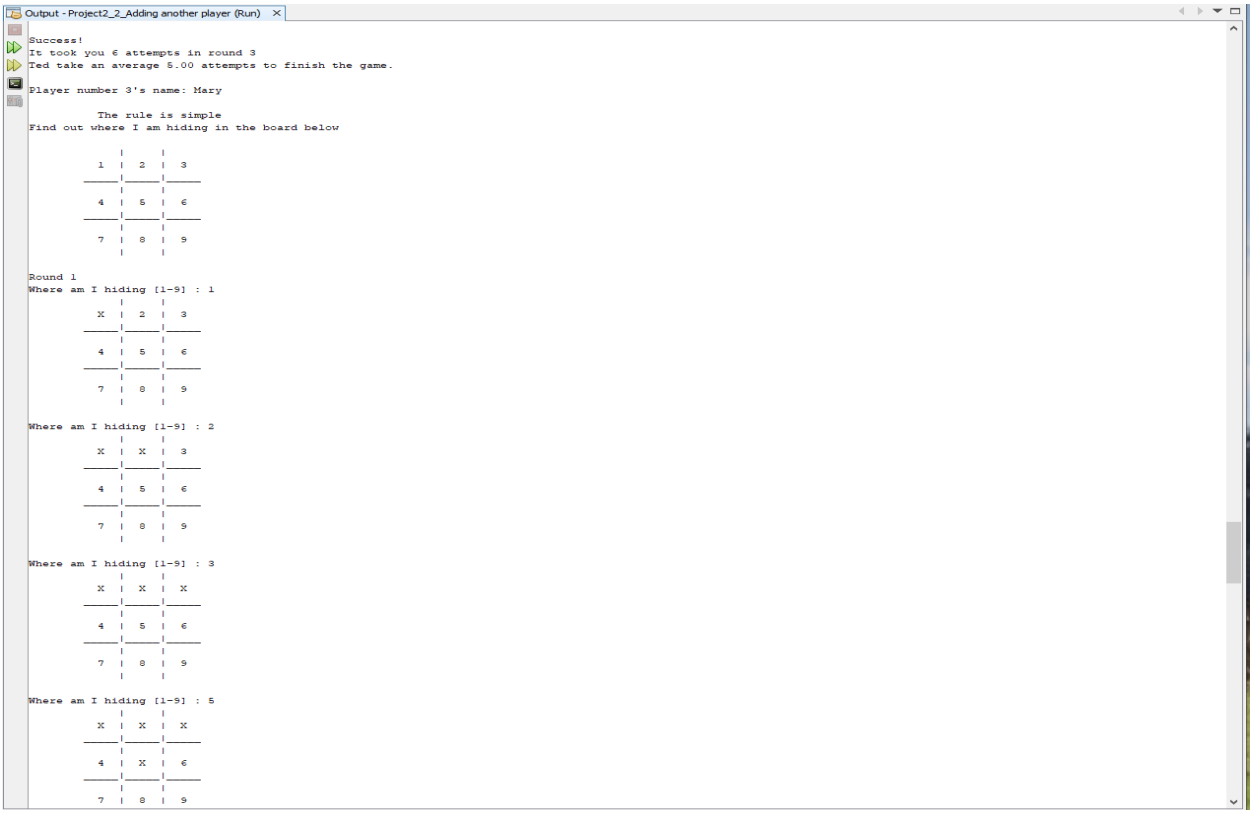



Output - Project2_2_Adding another player (Run)
X | 2 | X
4 | X | X
7 | 8 | 9
Where am I hiding [1-9] : 2
X | X | X
4 | X | X
7 | 8 | 9
Where am I hiding [1-9] : 4
X | X | X
X | X | X
7 | 8 | 9
Where am I hiding [1-9] : 7
X | X | X
X | X | X
X | 8 | 9
Where am I hiding [1-9] : 8
X | X | X
X | X | X
X | X | 9
Where am I hiding [1-9] : 9
X | X | X
X | X | X
X | X | 0

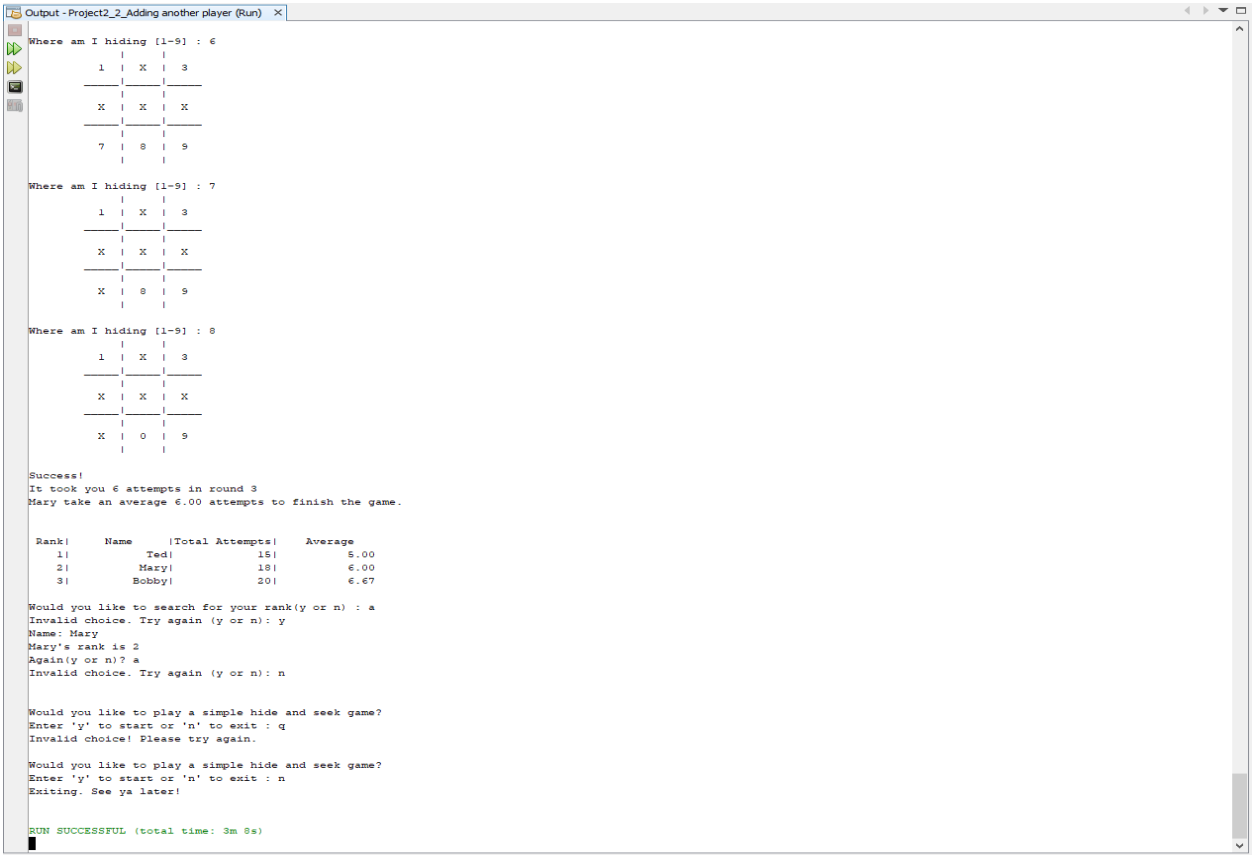

Output - Project2_2_Adding another player (Run)
X | X | 0
Success!
It took you 9 attempts in round 1
1 | 2 | 3
4 | 5 | 6
7 | 8 | 9
Round 2
Where am I hiding [1-9] : 1
X | 2 | 3
4 | 5 | 6
7 | 8 | 9
Where am I hiding [1-9] : 2
X | X | 3
4 | 5 | 6
7 | 8 | 9
Where am I hiding [1-9] : 3
X | X | X
4 | 5 | 6
7 | 8 | 9
Where am I hiding [1-9] : 4
X | X | X
X | 5 | 6
7 | 8 | 9
Where am I hiding [1-9] : 5


Output - Project2_2_Adding another player (Run)
Where am I hiding [1-9] : 6
Where am I hiding [1-9] : 7
Success!
It took you 6 attempts in round 3
Bobby take an average 6.67 attempts to finish the game.
Player number 2's name: Ted
The rule is simple
Find out where I am hiding in the board below
Round 1
Where am I hiding [1-9] : 1
Where am I hiding [1-9] : 2


Output - Project2_2_Adding another player (Run)
Where am I hiding [1-9] : 4
Success!
It took you 3 attempts in round 1
Round 2
Where am I hiding [1-9] : 2
Where am I hiding [1-9] : 4
Where am I hiding [1-9] : 0
Invalid input. Please try again.
Where am I hiding [1-9] : 10
Invalid input. Please try again.
Where am I hiding [1-9] : 3



```
Output - Project2_2_Adding another player (Run)
Success!
It took you 6 attempts in round 3
Ted take an average 5.00 attempts to finish the game.

Player number 3's name: Mary

          The rule is simple
Find out where I am hiding in the board below

          |     |
        1 |  2  |  3
      ____|_____|_____
          |     |
        4 |  5  |  6
      ____|_____|_____
          |     |
        7 |  8  |  9
          |     |


Round 1
Where am I hiding [1-9] : 1
          |     |
        X |  2  |  3
      ____|_____|_____
          |     |
        4 |  5  |  6
      ____|_____|_____
          |     |
        7 |  8  |  9
          |     |

Where am I hiding [1-9] : 2
          |     |
        X |  X  |  3
      ____|_____|_____
          |     |
        4 |  5  |  6
      ____|_____|_____
          |     |
        7 |  8  |  9
          |     |

Where am I hiding [1-9] : 3
          |     |
        X |  X  |  X
      ____|_____|_____
          |     |
        4 |  5  |  6
      ____|_____|_____
          |     |
        7 |  8  |  9
          |     |

Where am I hiding [1-9] : 5
          |     |
        X |  X  |  X
      ____|_____|_____
          |     |
        4 |  X  |  6
      ____|_____|_____
          |     |
        7 |  8  |  9
```



```
Output - Project2_2_Adding another player (Run)
Where am I hiding [1-9] : 6
          |     |
        1 |  X  |  3
      ____|_____|_____
          |     |
        X |  X  |  X
      ____|_____|_____
          |     |
        7 |  8  |  9
          |     |

Where am I hiding [1-9] : 7
          |     |
        1 |  X  |  3
      ____|_____|_____
          |     |
        X |  X  |  X
      ____|_____|_____
          |     |
        X |  8  |  9
          |     |

Where am I hiding [1-9] : 8
          |     |
        1 |  X  |  3
      ____|_____|_____
          |     |
        X |  X  |  X
      ____|_____|_____
          |     |
        X |  O  |  9
          |     |


Success!
It took you 6 attempts in round 3
Mary take an average 6.00 attempts to finish the game.


 Rank|     Name     |Total Attempts|    Average
    1|          Ted|            15|         5.00
    2|         Mary|            18|         6.00
    3|        Bobby|            20|         6.67

Would you like to search for your rank(y or n) : a
Invalid choice. Try again (y or n): y
Name: Mary
Mary's rank is 2
Again(y or n)? a
Invalid choice. Try again (y or n): n


Would you like to play a simple hide and seek game?
Enter 'y' to start or 'n' to exit : q
Invalid choice! Please try again.

Would you like to play a simple hide and seek game?
Enter 'y' to start or 'n' to exit : n
Exiting. See ya later!


RUN SUCCESSFUL (total time: 3m 8s)
```

Program coding:

```
/*
 * File:   main.cpp
 * Author: Triet Huynh
 * Created on October 14th, 2022
 * Purpose: Project 1_Simple Hide and Seek game in a 9 slots board
 */

//System Libraries
#include <iostream>  //I/O Library
#include <cstdlib>   //Random Function Library
#include <ctime>     //Time Library
#include <iomanip>   //Formatting Library
#include <cmath>     //math library
#include <fstream>   //file stream
#include <cctype>    //for tolower
#include "Arry.h"
#include "Stats.h"
using namespace std;

//User Libraries

//Global Constants, no Global Variables are allowed
//Math/Physics/Conversions/Higher Dimensions - i.e. PI, e, etc...

//Function Prototypes
void dplTble(const char [][3]);
float playGme(string,int&);
void playGme(int);
bool win(char[][3],int);
Stats *getInfo(int);
void bbleSort(int [], int);
void selSort(float [],string[], int);
void prntRnk(Stats *,int);
void del(Stats *);
void wrteBin(fstream &,Stats *);
Stats *readBin(fstream &);

//Execution Begins Here!
int main(int argc, char** argv) {
    //Set the random number seed
    srand(static_cast<unsigned int>(time(0)));

    //Declare Variables
    const unsigned short ROWS=3,
            COLS=3;
    int gmeType,        //single player or multiplayer
        numPlyer,       //number of players, 1 or 2
```

```
        atmpSum;    //Sum of attempts that player takes to finish the game
    char start;         //user choice to start or quit the game
    float atmpAvg;    //average number of attempts it takes for player to succeed per round
    string plyer;       //player's name

    //Initialize or input i.e. set variable values
    do{
       cout<<"Would you like to play a simple hide and seek game?"<<endl;
       cout<<"Enter 'y' to start or 'n' to exit : ";
       cin>>start;
    if(start=='y'||start=='Y'){
       cout<<endl<<"            Game starts!  "<<endl<<endl;
       cout<<"Enter 1 or 2:"<<endl
            <<"1. Single Player"<<endl
            <<"2. Multiplayer"<<endl;
       cin>>gmeType;
       //Input validation
       while(gmeType!=1&&gmeType!=2){
          cout<<"Invalid choice. Enter [1-2]."<<endl;
          cin>>gmeType;
       }
       if(gmeType==1){
          cout<<"Enter player's name: ";
          cin>>plyer;
          playGme(plyer,atmpSum);
       }else if(gmeType==2){
          cout<<"Input number of players: ";
          cin>>numPlyer;
          playGme(numPlyer);
    }else if(start=='n'||start=='N'){   //quits game
       cout<<"Exiting. See ya later!"<<endl<<endl;
       exit(0);
    }else{                          //invalid input
       cout<<"Invalid choice! Please try again."<<endl<<endl;
       }
    }
    }while(start!='n'&&start!='N');
    //Exit stage right or left!
    return 0;
}
//Function to display game table
void dplTble(const char table[][3]){
    cout<<"\t     |     |     "<<endl;
    cout<<"\t  "<<table[0][0]<<"  |  "<<table[0][1]<<"  |  "<<table[0][2]<<"  "<<endl;
    cout<<"\t_____|_____|_____"<<endl;
    cout<<"\t     |     |     "<<endl;
    cout<<"\t  "<<table[1][0]<<"  |  "<<table[1][1]<<"  |  "<<table[1][2]<<"  "<<endl;
    cout<<"\t_____|_____|_____"<<endl;
    cout<<"\t     |     |     "<<endl;
```

```
    cout<<"\t  "<<table[2][0]<<"  |  "<<table[2][1]<<"  |  "<<table[2][2]<<"  "<<endl;
    cout<<"\t     |     |     "<<endl<<endl;
}
//Single player game
float playGme(string plyer,int &atmpSum){
    const int ROWS=3,   //game board has 3 rows and 3 columns
          COLS=3;
    int choice,         //player's guess
          hide,       //where the object is hiding
          attmps,     //number of attempts it took for user to find the right spot
          temp,       //temporary value to hold number of attempts in each round
          rounds;     //3 rounds in total
    float atmpAvg;    //average number of attempts it takes for player to succeed per round
       cout<<endl<<"          The rule is simple"<<endl;
       cout<<"Find out where I am hiding in the board below"<<endl<<endl;
       ofstream outputFile(plyer);
       atmpSum=0;
       //start game from round 1->3
       for(rounds=0;rounds<3;rounds++){
          char table[ROWS][COLS]={{'1','2','3'},{'4','5','6'},{'7','8','9'}};   //game's board
          hide=(rand()%9)+1; //generate a random number as hiding slot
          attmps=0,temp=0;
          //display table
          dplTble(table);
          cout<<"Round "<<rounds+1<<endl;
          do{
             cout<<"Where am I hiding [1-9] : ";
             cin>>choice;
             //input validation
             if(choice<1||choice>9)
                cout<<"Invalid input. Please try again."<<endl;
             else{
             attmps++;
             switch (choice){
                case 1:{
                   if(choice==hide)
                   {
                      table[0][0]='0';
                      //display updated table
                      dplTble(table);
                      break;
                   }else
                   {
                      table[0][0]='X';
                      //display updated table
                      dplTble(table);
                      break;
                   }
                }
```

```
case 2:{
   if(choice==hide)
      {
         table[0][1]='0';
         dplTble(table);
         break;
      }else
      {
         table[0][1]='X';
         dplTble(table);
         break;
      }
   }
case 3:{
   if(choice==hide)
      {
         table[0][2]='0';
         dplTble(table);
         break;
      }else
      {
         table[0][2]='X';
         dplTble(table);
         break;
      }
}
case 4:{
   if(choice==hide)
      {
      table[1][0]='0';
         dplTble(table);
         break;
      }else
      {
         table[1][0]='X';
         dplTble(table);
         break;
      }
}
case 5:{
   if(choice==hide)
      {
         table[1][1]='0';
         dplTble(table);
         break;
      }else
      {
         table[1][1]='X';
         dplTble(table);
```

```
            break;
        }
}
case 6:{
    if(choice==hide)
        {
            table[1][2]='0';
            dplTble(table);
            break;
        }else
        {
            table[1][2]='X';
            dplTble(table);
            break;
        }
}
case 7:{
    if(choice==hide)
        {
            table[2][0]='0';
            dplTble(table);
            break;
        }else
        {
            table[2][0]='X';
            dplTble(table);
            break;
        }
}
case 8:{
    if(choice==hide)
        {
            table[2][1]='0';
            dplTble(table);
            break;
        }else
        {
            table[2][1]='X';
            dplTble(table);
            break;
        }
}
case 9:{
    if(choice==hide)
        {
            table[2][2]='0';
            dplTble(table);
            break;
        }else
```

```
                {
                   table[2][2]='X';
                   dplTble(table);
                   break;
                }
             }
             }
             }
          }while(choice!=hide);
          if(win(table,COLS)==true){
          cout<<"Success!"<<endl;
          cout<<"It took you "<<attmps<<" attempts in round "<<rounds+1<<endl;
          }
          outputFile<<setw(6)<<attmps<<endl;   //save player's attempts in file
       }
       outputFile.close();
       ifstream inputFile;
       inputFile.open(plyer);  //create a file with player's name and store attempts for record
       while(inputFile>>temp){
          atmpSum+=temp;
       }
       inputFile.close();
       atmpAvg=abs(static_cast<float>(atmpSum)/3.0);
       cout<<fixed<<showpoint<<setprecision(2);
       cout<<plyer<<" take an average "<<atmpAvg<<" attempts to finish the game."<<endl<<endl;
       return atmpAvg;
}
//check for winning status
bool win(char table[][3],int cols){
   enum Truth{FALSE,TRUE};
   Truth result=FALSE;
   for(int i=0;i<3;i++){
      for(int j=0;j<cols;j++){
         if(table[i][j]=='0')
            result=TRUE;
      }
   }
   return result;
}
//Multiplayer game
void playGme(int numPlyr=1){
   string nmeSrch;       //player's name to search for rank
   char search,          //whether player wants to search for their rank
        again;         //search again? yes or no
   fstream binFile;
   int MAX_PLYRS=20;      //maximum 20 players
   Stats players[MAX_PLYRS];
   binFile.open("data.bin",ios::in|ios::out|ios::binary|ios::trunc);
   //dynamically allocate structure pointers
```

```
    Stats *player=nullptr;
    Stats *copy=nullptr;
    //perform game with function for each player
    player=getInfo(numPlyr);
    //Sort the arrays
    bbleSort(player->sum,numPlyr);
    selSort(player->avg.data,player->names,numPlyr);
    //Write to and from binary file
    wrteBin(binFile,player);
    copy=readBin(binFile);
    //Display the ranking board
    prntRnk(copy,numPlyr);
    cout<<endl<<endl;
    //reverse and delete dynamically allocated memory
    del(player);
    delete copy;
    player=nullptr;
    copy=nullptr;
    //close file
    binFile.close();
}

//Perform game and gather scores for each player
Stats *getInfo(int numPlyr){
    string plyNme;   //player's name
    int atmpSum;           //sum of player attempts after 3 rounds
    float atmpAvg;         //average attempts to finish 3 rounds
    //dynamically allocate structure array
    Stats *a=new Stats;
    a->players=numPlyr;
    a->names=new string[a->players];
    a->sum=new int[a->players];
    a->avg.size=a->players;
    a->avg.data=new float[a->avg.size];
    //perform game until last player is done and fill in names, total score and avg score
    for(int i=0;i<numPlyr;i++){
        cout<<"Player number "<<i+1<<"'s name: ";
        cin>>plyNme;
        a->names[i]=plyNme;
        atmpAvg=playGme(plyNme,atmpSum);
        a->avg.data[i]=atmpAvg;
        a->sum[i]=atmpSum;
    }
    return a;
}

//Bubble sort
void bbleSort(int num[], int size){
    for(int maxEle=size-1;maxEle>0;maxEle--){
```

```
        for(int indx=0;indx<maxEle;indx++){
            //if first value is greater than second value, swap them
            if(num[indx]>num[indx+1]){
                int temp=num[indx];
                num[indx]=num[indx+1];
                num[indx+1]=temp;
            }
        }
    }
}
//Dual sort with array names and array average using selection sort
void selSort(float avg[],string names[], int size){
    float minVal;
    int minIndx;
    for(int start=0;start<(size-1);start++){
        minIndx=start;
        minVal=avg[start];
        for(int indx=start+1;indx<size;indx++){
            if(avg[indx]<minVal){
                minVal=avg[indx];
                minIndx=indx;
            }
        }
        //if value of index indx in first array is less than min value, swap them
        float flTemp=avg[minIndx];
        avg[minIndx]=avg[start];
        avg[start]=flTemp;
        //swap elements in second array accordingly
        string strTemp=names[minIndx];
        names[minIndx]=names[start];
        names[start]=strTemp;
    }
}
//Print the ranking board
void prntRnk(Stats *player,int numPlyr){
    cout<<endl<<" Rank|     Name     |Total Attempts|    Average   "<<endl;
    for(int i=0;i<numPlyr;i++){
        cout<<setw(5)<<i+1<<"|"<<setw(14)<<player->names[i]<<"|"
                <<setw(14)<<player->sum[i]<<"|"
                <<setw(14)<<player->avg.data[i]<<endl;
    }
}
//delete dynamically allocated memory
void del(Stats *a){
    delete [] a->avg.data;
    delete [] a->names;
    delete [] a->sum;
    delete a;
}
```

```
//Write structure to binary file
void wrteBin(fstream &out,Stats *a){
   out.write(reinterpret_cast<char *>(&a->players),sizeof(int));
   out.write(reinterpret_cast<char *>(a->names),a->players*sizeof(string));
   out.write(reinterpret_cast<char *>(a->sum),a->players*sizeof(int));
   out.write(reinterpret_cast<char *>(&a->avg.size),sizeof(int));
   out.write(reinterpret_cast<char *>(a->avg.data),a->avg.size*sizeof(float));
}
//Read data structure from binary file
Stats *readBin(fstream &in){
   Stats *a=new Stats;
   long cursor=0L;
   //read data to structure
   in.seekg(cursor,ios::beg);
   in.read(reinterpret_cast<char *>(&a->players),sizeof(int));
   a->names=new string[a->players];
   in.read(reinterpret_cast<char *>(a->names),a->players*sizeof(string));
   a->sum=new int[a->players];
   in.read(reinterpret_cast<char *>(a->sum),a->players*sizeof(int));
   in.read(reinterpret_cast<char *>(&a->avg.size),sizeof(int));
   a->avg.data=new float[a->avg.size];
   in.read(reinterpret_cast<char *>(a->avg.data),a->avg.size*sizeof(float));
   return a;
}
```