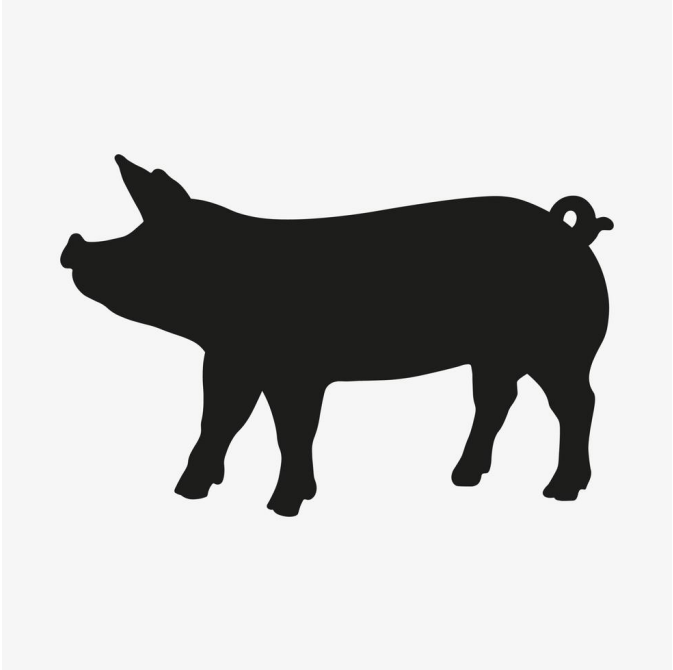
Project 2:

Game of Pig



Name: Venus Curtner

Date: 12/16/2022

CIS-17

Table of Contents

Introduction:

Game: **Pig**

The game I chose to model is called: Pig. I chose this game because you can play it anywhere if you have 1 die, or something you can make into a die. The rules of the game ‘Pig’ are pretty universal, and it is an established game played outside of my own circle. The rules are relatively simple for the game, my program only deviates from the rules by changing it from 2+ players to 1 player against a robot. The player preselects the difficulty of the bot, which affects its potential scores. The rules are outputted to the player at the start of the game from a text file matching ***exactly*** to the following:

```
**********************************************************
********************Game of Pig***************************
**********************************************************
Are you greedier than a pig? ...

The goal is to reach 100 before the robot does. On your
turn you may roll as many times as you would like. Each
time you roll, add the value on the die to your TURN'S
total. You may stop rolling whenever you like, then add
your TURN'S total to your OVERALL total for the game.
However, should you roll a 1 during your turn, all of
the TURN'S progress is reset to 0, and your turn is over.
In other words, should you roll a 1 during your turn,
you are the PIGGY, and and a PIGGY makes no progress for
the remainder of their turn.

You are playing against a robot named 'Robot'...or?
something else...see if you can find its SECRET name...
When your turn is over, the robot adds to its score,
points will be based on pre-selected difficulty mode.
a robot is not greedy, so it will only roll once and
will never be a PIGGY.

To win this game you can either be brave...or weak...
What will you be?

**********************************************************
********************Let's Play****************************
**********************************************************
```

File Summary:

Instructions are read in from instructions.txt located in final version folder
Winners are recorded into a file scores.txt also located in final version folder
Players Guess.txt and RobotsRating.txt are used to demonstrate a principle
Robotscores.txt and playerscores.txt record both robot and players turns throughout the game.
Various statistics about the previous game are displayed in data.txt located in the final version folder.
Example inputs inside final version folder

Total lines in project: **657**

PseudoCode:

Open File
    Print Instructions
Close File

Enter getnames
    If name is Dr Lehr or Mark Lehr
        Output "is my teacher"
    If not
        Output "is not my teacher"
    If name is Robot
        Robot name is Droid
    If not
        Robot name is Robot

Enter Difficulty Rating
    If 1
        Robots array of potential scores becomes: 2
    If 2
        Robots array of potential scores becomes: 2, 4
    If 3
        Robots array of potential scores becomes: 4, 6

turn total reset

Roll
    If die lands 1
        reset turn progress
    If die does not land 1
        Add to turn total, record
            Ask for reroll
        If Y or y
            Go back to 'Roll'
        If N or n
            Record Score exit turn

Robot scores

If neither player or robot have reached 100
    Go back to turn total reset
If they have,
    Exit roll
    Game over
    Record all scores
    Determine Statistics based on results of game
    Output Scores

Return 0
End

FlowChart: Linked in folder

GitHub Repository: Linked in folder
(Contains Previous Versions)

Cross Reference Sheet: Linked in folder

7

Final Code:

PigSetup.h

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   PigSetup.h
 * Author: wyatt
 *
 * Created on December 16, 2022, 6:10 PM
 */

#ifndef PIGSETUP_H
#define PIGSETUP_H

#include <cstring>
#include <string>
#include <iostream>
using namespace std;
class Pers
{

public:


    string name="default";
    string title="unranked";
};
class PlayInf{
protected:
    string win;

public:
```

```
    int* chDiff;



    int bravery;
    int turn;
    int NumYes;
    int NumNos;
    int Numtimes;
    float YesPerc;
    float NoPerc;
    bool result;
    Pers personal;

    void setwin(string);
    string getwin ()const;
};



class message{
public:
    message();
    message (bool z);
    ~message();
};

class points{
public:

    int scores[400];
     int numturn=0;
};

class guess {
private:
    int x;
```

```
public:
    guess(int x1)
    {
        x = x1;

    }

    // Copy constructor
    guess(const guess& p1)
    {
        x = p1.x;
    }
    int getX() { return x; }

};

class Totality {
private:
    int xS;
public:
    Totality(int r = 0) {xS= r;}

    // This is automatically called when '+' is used with
    // between the objects
    Totality operator + (Totality const &obj) {
         Totality totalS;
         totalS.xS= xS + obj.xS;

         return totalS;
    }
    void print() { cout<<"The two scores combined
equal:"<<xS<<endl; }
};
#endif /* PIGSETUP_H */
```

PigSetup.cpp

```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   PigSetup.cpp
 * Author: wyatt
 *
 * Created on December 16, 2022, 6:48 PM
 */

#include <cstdlib>
#include "PigSetup.h"
#include <iostream>
using namespace std;



void PlayInf::setwin(string w){
    win=w;
}

string PlayInf::getwin ()const{
    return win;
}
message::message(){
    cout<<"Welcome to my Project 2! This is a constructor in my
Project. Hope you enjoy the game."<<endl;
}


message::message(bool z){
    if (z==true){
        cout<<"Hope you had fun!"<<endl;
    }
}
```

```
message::~message(){
    cout<<"this is my destructor. by the way...Is it groundhogs
day?"<<endl;
}
```

Main

```
/*
 * File:   main.cpp
 * Author: Venus Curtner
 * Last Updated on: May 13th, 9:03 PM
 * Purpose: Final Version of my Project 1
 */
//System Libraries
#include <iostream>  //Input/Output Library
#include <cstdlib> // Random Number Generation
#include <ctime> // Time Clock Manipulation
#include <iomanip> //Input Output Formatting
#include <string> // String Library
#include <cstring>
#include <cmath> // Math Libary
#include <cctype>
#include <fstream> // File Library
#include<bits/stdc++.h>

#include "PigSetup.h"


using namespace std;

//User Libraries


enum numberz {zero, one, two, three};
//Global Constants, no Global Variables are allowed

//Math/Physics/Conversions/Higher Dimensions - i.e. PI, e,
etc...
const int PRCNT = 100; // Percentage conversion, constant
```

```
//Function Prototypes
struct PlayInf GetBrave (int);
void RecData (fstream& Data, PlayInf Person);
void ToFile(string filenm,PlayInf Person);
void PlayLog(string filenm,PlayInf Person);
bool checkwin (PlayInf Person);
void BraveRate(PlayInf Person);
string getnames(string* robname = NULL);
void RecChoice (string ,string , points x);
void RecGuess (string file, string tag, guess x);
//Execution Begins Here!

template <typename V> V getMax(V x, V y)
{
    if (x==y){
        throw "Equal Numbers Accepted into getMax";
    }
    return (x > y) ? x : y;
}



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

    message hi;
//Set Random Number Seed
srand(static_cast<unsigned int>(time(0))); // seed for randoms

PlayInf Person;
points Player;
points Robot;



//Declare Variables
string P1Nm;
  string RbNm;



string winner; // winner is:
```

```
int turn; //the number of turns taken by p1
int robot; // robot player plays against



int result; //Result of turn
int Ntms, WNY, WNN; //Number of times p1 presses y or Y
float YesP, NoP; //Yes percentage + No percentage
float  FWNY, FWNN; //Float value of WNY + WNN
bool WinYt; //Has p1 won yet?
bool canRl, iswmp; // cancel roll?
bool cancel; // cancel trnTl?
        int ScrTurn;

int brave; // Numeric Recording of Bravery value
int die; // stores die, later randomized as [1-6]
int potent; // potential to add if roll != 1
int NowScr; // score now equals
int trnTl; //total accumulated on this turn
unsigned char again; // players choice to roll again

string file ="Data.txt";
string plyrC="playerscores.txt";
string roboC="robotscores.txt";
string file2= "PlayerLog.txt";
string plyrG="players guess.txt";
string roboG="robotsrating.txt";
//Initialize Variables
brave=0; // initialize bravery rating to 0
NowScr=0; // initialize starting score to 0
robot=0; // initialize robot's score to 0
turn=0; // initialize turn to 0



Ntms=WNY=WNN=0; //initialize number of times winner pressed
anything, yes, and no to 0
WinYt=false; // initialize player's winning or not to false
potent=0; // initialize potential per turn to 0 (repeat in loop)
trnTl=0; // initialize turn's collective score to 0
```

```
cancel=false; // initialize cancel re-roll to 0
iswmp=false;
again='D'; // initialize again input to 'D' for default


//Initialize Variables (Text Lines)
string Rls1="PIGGY!!! That sucks...next player's turn.";//If
player rolls 1
string RlsNO="You're safe for now...add to turn total: ";//If
player does not roll 1
string QAg="Wanna go again..?, press y or n."; //Question
prompting again
string EnNm="Enter Player 1's Name: "; //Enter Name Prompt
string RbTxt="You are playing against a "; //Robot prompt
string tagP="You guessed that the Robot would score:";
string tagR="It was predicted that the Robot would score:";
//Introduce Game + Explain Rules & Prompts


  ifstream instr ("instructions.txt", ifstream::binary);
  if (instr) {
    // get length of instructions:
    instr.seekg (0, instr.end);
    int length = instr.tellg();
    instr.seekg (0, instr.beg);

    // allocate memory w smart pointer:
    char * buffer = new char [length];

    // read data as a block:
    instr.read (buffer,length);

    instr.close();

    // print instructions
    cout.write (buffer,length);

    delete[] buffer;
  }

//Enter player name
```

```
cout<<endl;
cout<<EnNm; // output entered name line
P1Nm = getnames(&RbNm);
Person.personal.name=P1Nm;
cout<<RbTxt<<RbNm<<endl;  // output robot text line


 int v=0,n;
  int * p;
  while (v>three || v<one){
  cout << "What Difficulty Rating would you like? Enter a Number
1-3:";
  cin>>v;
  }
  p= new (nothrow) int[v];
  if (p == nullptr){
    cout << "Error: memory could not be allocated";
  }
  if (v==1){
      cout<<"Difficulty Rating: Easy";
      cout<<endl;
  }
  if (v==2){
      cout<<"Difficulty Rating: Medium";
      cout<<endl;
  }
  if (v==3){
      cout<<"Difficulty Rating: Hard";
      cout<<endl;

  }

  int predict=0;

  while (predict<1 || predict>150){
  cout<<"How many points do you think the robot will
get?"<<endl;
  cout<<"Potential Score of (1-150):";
  cin>>predict;
  }
  guess p1(predict); // Normal constructor is called here
```

```
  guess r1 = p1; // Copy constructor is called here

  int robdie=0;
  for (int i=0; i<v; i++){
      robdie+=2;
      p[i]=robdie;

  }




  ScrTurn=0;
do{ // do everything inside as long as:

    cout<<"You are about to roll..."<<endl; // output statement
for enter do while
    trnTl=0;
    do{ // Will Keep Rolling Die as long as: Roll has not been
cancelled & re-roll input is yes

cout<<"++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++"<<endl;
        //Roll Die

            cout<<"Rolling..."<<endl; // output statement
            turn++;
            die=(rand()%6)+1; // roll [1-6]
            cout<<"You landed a :"<<die<<endl; // output
statement + result

            cout<<endl; // output end line

        //is die 1 or not 1
        if (die==1){ //you rolled a 1

            trnTl=0; //turn progress resets
            canRl=false; //cannot roll again
            cout<<Rls1<<endl; // text line
            cancel=true; //cancel the addition of trnTl
```

```
        } //leave conditions for die = 1

        else if (die!=1){ // you did not roll a 1 [2-6]

            trnTl+=die; //add roll to total for turn
            canRl=true; //can roll again
            cout<<RlsNO<<die<<endl; // text line
            cancel=false; // do not cancel the addition of trnTl
            cout<<"Currently you have accumulated:
"<<trnTl<<endl; // output statement + turn's total + end line
            cout<<"Before this turn you had: "<<NowScr<<endl; //
output statement + player's score + end line
            potent=(NowScr+trnTl); //potential is added to p1's
score
        } // leave conditions for die != 1

            cout<<"Now you Have: "<<potent<<endl; //potential
amount if 1 does not get rolled
            cout<<endl;

        if ((canRl==true)&&(potent<100)){ //Can player re-roll?
+ cut off re-roll at 100
            // Prompt ReRoll
            cout<<QAg<<endl; //Text line prompting choice

           do{ //Input Validation for Re-Roll Choice
                cout<<"Make sure you type: Y or N capitalization
does not matter."<<endl;
                cin>>again; // input choice for again
                if (toupper(again)=='Y'){WNY++;Ntms++;} // if y
or Y is pressed add one to WNY and Ntms
                if (toupper(again)=='N'){WNN++;Ntms++;} // if n
or N is pressed add one to WNN and Ntms


            }while
((again=='D')||((toupper(again)!='N')&&(toupper(again)!='Y')));



        }//re-roll choice allowed and confirmed
```

```
        } while
(((again=='Y')||(again=='y'))&&(canRl!=false)&&(potent<=100));
//continue as long as again is yes and roll hasn't been canceled


        ScrTurn++;
        NowScr+=trnTl; // Player's score collects value of turn
total
        cout<<"Testing, on turn:"<<ScrTurn<<" and the array is
at:"<<(ScrTurn-1)<<endl;
        Player.scores[ScrTurn-1]=NowScr;
        cout<<"Recorded:"<<Player.scores[ScrTurn-1]<<endl;



cout<<"************************************************"<<endl;
//output statement + end line




        cout<<RbNm<<" adds to total"<<endl; // output statement
+ end line
        int robscr,robroll=0;

        robroll=(rand()%v)+1;
        robscr=p[robroll-1];
        robot+=robscr;
        Robot.scores[Robot.numturn]=robot;
        Robot.numturn++;
        cout<<RbNm<<"'s score is at: "<<robot<<endl;


        //Game Over?
        if (NowScr>=100){
           WinYt=true;
        } // if the score reaches 100 WinYt=true;
```

```
}while(WinYt==false && robot<100 ); // as long as WinYt equals
false and robot has not reached 100
Person.result=WinYt;
switch (WinYt){ // switch these cases based on the value of
WinYt
    case false:winner=RbNm;break; // in the case that WinYt is
false winner is "Robot"
    case true: winner=Person.personal.name;break;  // in the
case that WinYt is true then winner is Player
}
//Game Over
message bye(WinYt);
Player.numturn=ScrTurn;
  delete[] p;
FWNY=WNY; // Mix data types for float calculations
FWNN=WNN;
YesP=((FWNY/Ntms)*PRCNT); // Percentages of times player said
yes equals the number of times they pressed yes
//divided by the number of times they said anything, times 100
NoP=((FWNN/Ntms)*PRCNT); // Percentages of times player said no
equals the number of times they pressed no
//divided by the number of times they said anything, times 100

//store info in structure for use with printing

Person.setwin(winner);
Person.turn=turn;
Person.NumYes=WNY;
Person.NumNos=WNN;
Person.Numtimes=Ntms;
Person.YesPerc=YesP;
Person.NoPerc=NoP;
//
FWNY=WNY; // Mix data types for float calculations
FWNN=WNN;

cout<<fixed<<setprecision(2)<<showpoint; //output formatting
```

```
if((YesP>=50.00)&&(YesP<60.00)){brave=1;} // if the percentage
of times player pressed yes is above:50-60 bravery rating is:1
if((YesP>=60.00)&&(YesP<70.00)){brave=2;} // if the percentage
of times player pressed yes is above:60-70 bravery rating is:2
if((YesP>=70.00)&&(YesP<80.00)){brave=3;} // if the percentage
of times player pressed yes is above:70-80 bravery rating is:3
if((YesP>=80.00)&&(YesP<90.00)){brave=4;} // if the percentage
of times player pressed yes is above:80-90 bravery rating is:4
if(YesP>=90.00)                {brave=5;} // if the percentage
of times player pressed yes is above:90    bravery rating is:5

Person.bravery=brave;
```

```
cout<<"bravery Rating: ";// output statement
for (int i = 0; i <= (brave) ; i++){ //output "*", as many times
as the value of brave
    cout<<"*";// output "*"
}
cout<<endl; // output end line
cout<<"        out of: *****"<<endl; // output statement + end
line
cout<<"because you said yes around "; // output statement
cout<<floor(YesP)<<"% of the time."<<endl; // output rounded
down value of YesP + statement + end line
while((YesP<50.00) && (iswmp==false)){
    cout<<"whimp..."<<endl;
    iswmp=true;
}
//Store Winner Name in main.cpp
ofstream recwin; // record wins here
recwin.open("scores.txt", ios::app); // open scores.txt save our
changes and leave them each run
```

```
recwin<<winner<<" won."<<endl; // output winner + statement +
end line into file
recwin.close(); // close file


PlayInf Initials;
Initials=GetBrave(brave);
cout<<Person.personal.name<<" is
"<<Initials.personal.title<<endl;
ToFile(file, Person);// Records Winner's Stats in a file.
RecChoice(plyrC, P1Nm, Player);
RecChoice(roboC, RbNm, Robot);
RecGuess ( plyrG, tagP, p1);
RecGuess ( roboG, tagR, r1);
//Exit Program
int realG;
realG=getMax<int>(predict,robot);
if (realG==predict){
    cout<<"The robot scored less than you thought."<<endl;
}
if (realG==robot){
    cout<<"The robot scored more than you thought."<<endl;
}
if (robot=predict){
    cout<<"Perfect Guess! Robot's score was as you'd
thought."<<endl;
}

Totality xS1(NowScr), xS2(robot);
    Totality xS3 = xS1 + xS2;
    xS3.print();
return 0;
}

void ToFile(string filenm,PlayInf Person){
cout<<"Writing to the file:"<<endl;

ofstream ofile;
ofile.open(filenm.c_str(),ios::binary|ios::out);
```

```
ofile<<"Game Finished!"<<endl; // output statement + end line
ofile<<"The Winner is: "<<Person.getwin()<<endl; // output
statement + winner + end line
ofile<<Person.personal.name<<" played for "<<Person.turn<<"
turns."<<endl; // output winner + statement + turn + statement +
end line
ofile<<Person.personal.name<<" said yes to re-roll
"<<Person.NumYes<<" times."<<endl; // output winner + statement
+ WNY + statement + end line
ofile<<"and no to re-roll "<<Person.NumNos<<" times."<<endl; //
output statement + WNN + statement + end line
ofile<<"that makes "<<Person.Numtimes<<" times total."<<endl;
// output statement + Ntms + statement + end line
ofile<<"That means you said yes: "<<Person.YesPerc<<"% of the
time."<<endl; // output statement + YesP + statement + end line
ofile<<"and no: "<<Person.NoPerc<<"% of the time."<<endl; //
output statement + NoP + statement + end line




ofile.close();
}
void RecGuess (string file, string tag, guess x){
   ofstream ofile;
ofile.open(file.c_str(),ios::binary|ios::out);

ofile<<tag<<x.getX()<<endl;

ofile.close();
}
void RecChoice (string filenm,string name, points x){

 ofstream ofile;
ofile.open(filenm.c_str(),ios::binary|ios::out);



    int amt;
amt=x.numturn;
```

```
for (int i=0; i<amt; i++){
    ofile<<"On turn #"<<(i+1)<<":";
    ofile<<x.scores[i]<<endl;
}

ofile.close();
}

struct PlayInf GetBrave (int x){
     struct PlayInf plyr;

     if (x=1){
         plyr.personal.title="scaredy cat";
     }
     if (x=2){
         plyr.personal.title="a whimp";
     }
     if (x=3){
         plyr.personal.title="lame";
     }
     if (x=4){
         plyr.personal.title=" a risk taker";
     }
     if (x=5){
         plyr.personal.title="brave";
     }

     return plyr;
}

string getnames(string* robname){
  string P1Nm ;
  string RbNm;

getline (cin,P1Nm); // input player name

char arrNm[P1Nm.length() + 1];

     strcpy(arrNm, P1Nm.c_str());
        char * strptr=nullptr;
        strptr=strstr(arrNm,"Mark Lehr");
```

```
            if (strptr!=nullptr){
            cout<<P1Nm<<" is my teacher"<<endl;
        }
        strptr=strstr(arrNm,"Dr. Lehr");
            if (strptr!=nullptr){
            cout<<P1Nm<<" is my teacher"<<endl;
        }

        if (strptr==nullptr){
            cout<<P1Nm<<" is not my teacher"<<endl;
        }
```

```
cout<<endl; // output end line
RbNm= (P1Nm=="Robot")? "Droid" : "Robot"; // RbNm = Droid if
P1Nm= Robot, else = Robot

  if(robname != NULL) {
    *robname = RbNm;
  }
  return P1Nm;
}
```