Heist Dice

Project #1

by Norman Lee
for CSC-5
Date: 7/20/15

-Introduction-

This is the first project for my CSC5 class. Heist Dice is inspired from a game called Zombie Dice. You play a team of robbers who try to get as much loot as possible, while trying to avoid getting strikes on your record. Like Zombie Dice everything is determined by weighted, colored dice; the dice representing a *heist* you are trying to pull off. The different colors of dice indicate how difficult the job will be. Each round represents a member of your crew. Your opponent is a rival robbery gang. Compete to find out who is the best!

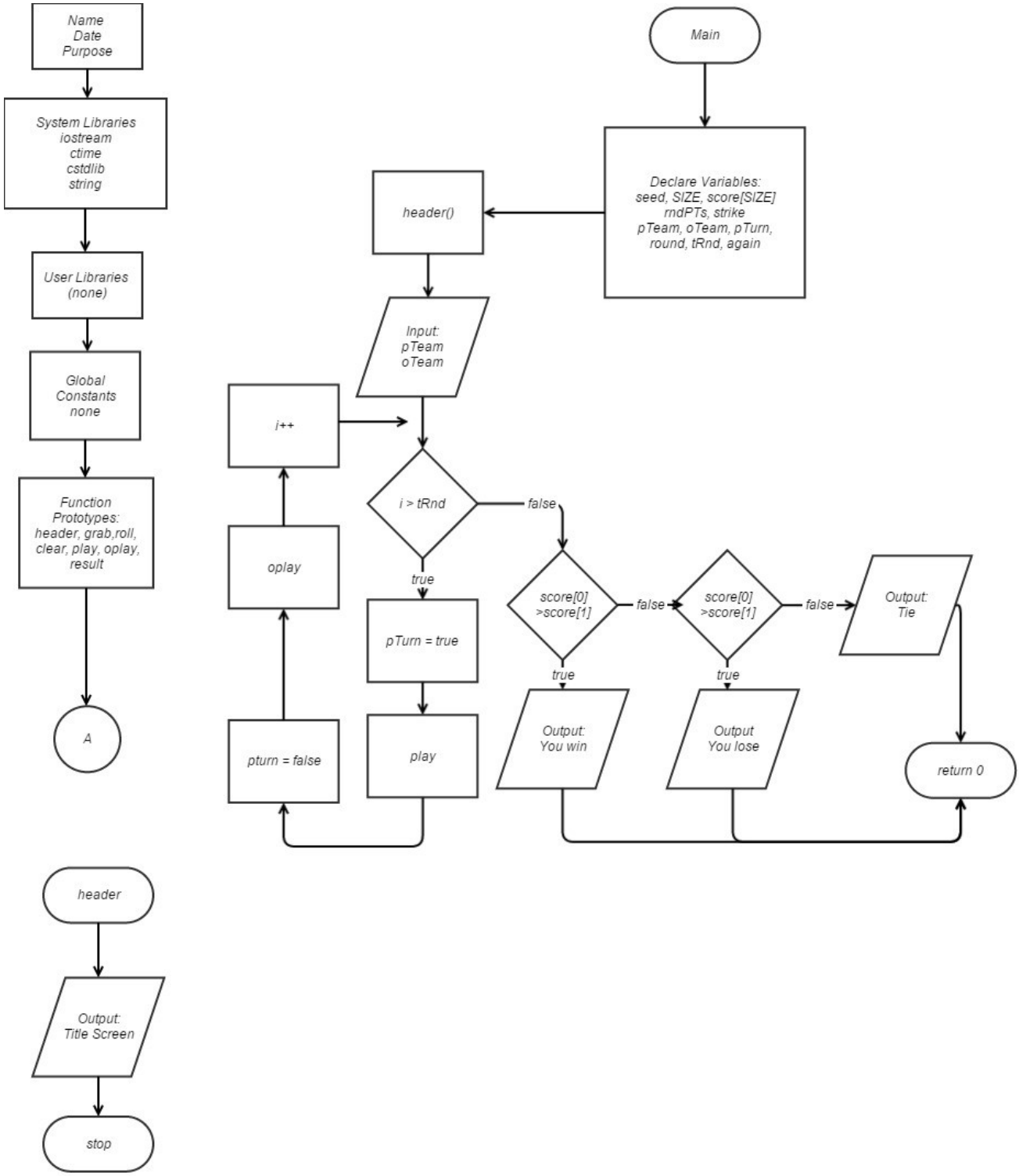
-Instructions-

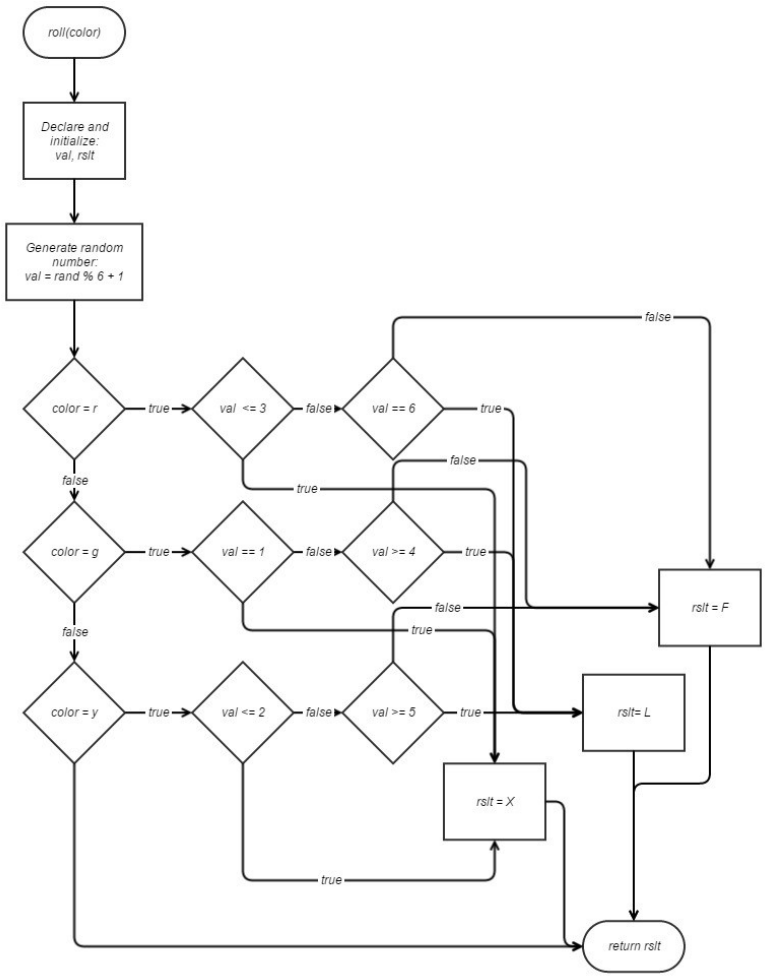
In Heist Dice, you roll to see if you get loot, escape without harm or get caught and end up with a strike on your record. Your first action is to pick a random dice of out a bag. There are 3 different colors of dice, each weighted differently. Green dice have the highest number of sides with loot, Red dice have the highest number of strikes, and Yellow dice have even odds for all outcomes. Every round you roll the dice until you either get 3 strikes or choose to stop. If you get 3 strikes, your crew member faces life in prison and is forced to give up all the loot to get a plea deal. You take turns rolling with your opponent for 5 rounds and whoever has the most points at the end wins.

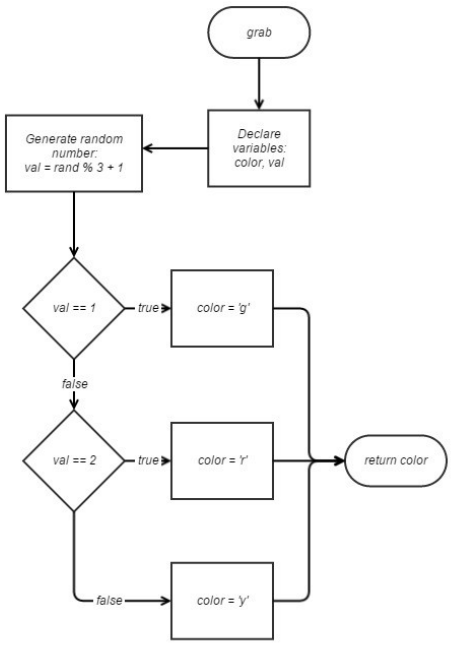
-Design Details-

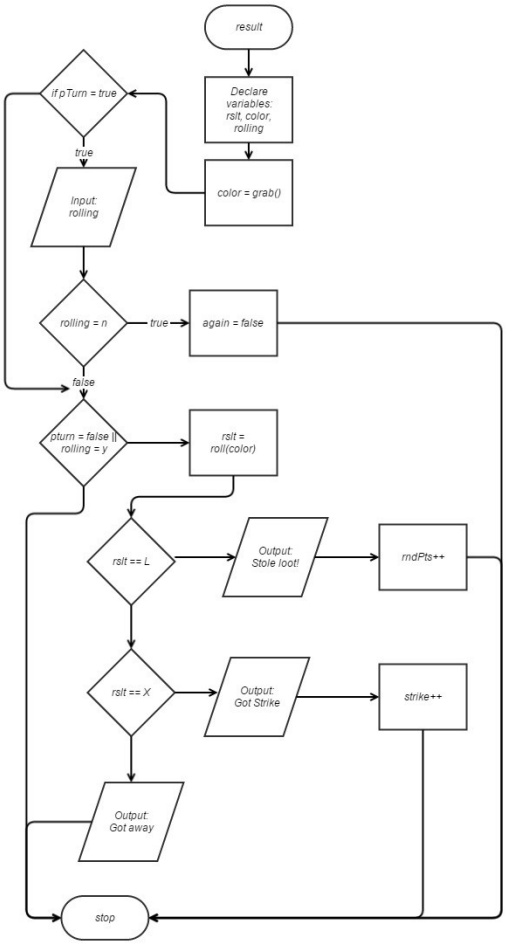
My strategy for coding this game was to first start by programing the dice. Being as this is an original game, I could change the rules during the process, so the next step was setting up a tentative system so I could test various ways of playing the game. I originally had it so, like Zombie Dice, you chose to continue after each roll. But after testing it, the system felt too random and your choices didn't really matter. Zombie Dice has some elements to it that give it more strategy; like the fact you roll 3 dice at a time and keep dice that roll footprints. I felt that rolling 1 dice portrayed the theme of my game better as each die roll would be like a “job” the character would do. It didn't make sense to have 3 different heists run concurrently. My solution was to make the choice of rolling happen when you get a random color for your dice. The draw back in having not set rules from the start is I ended up spending a lot of time shuffling code between different functions and trying to make them work the way that made the most sense.

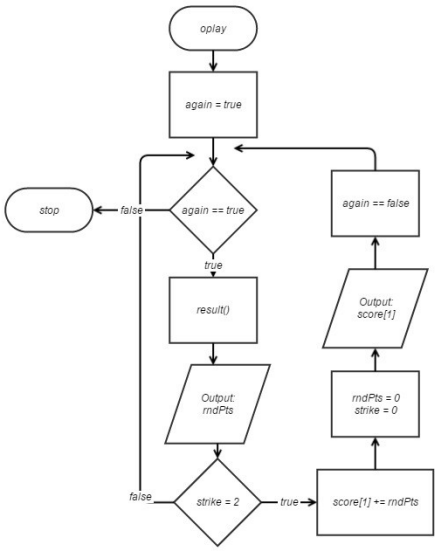
-Flow Chart-

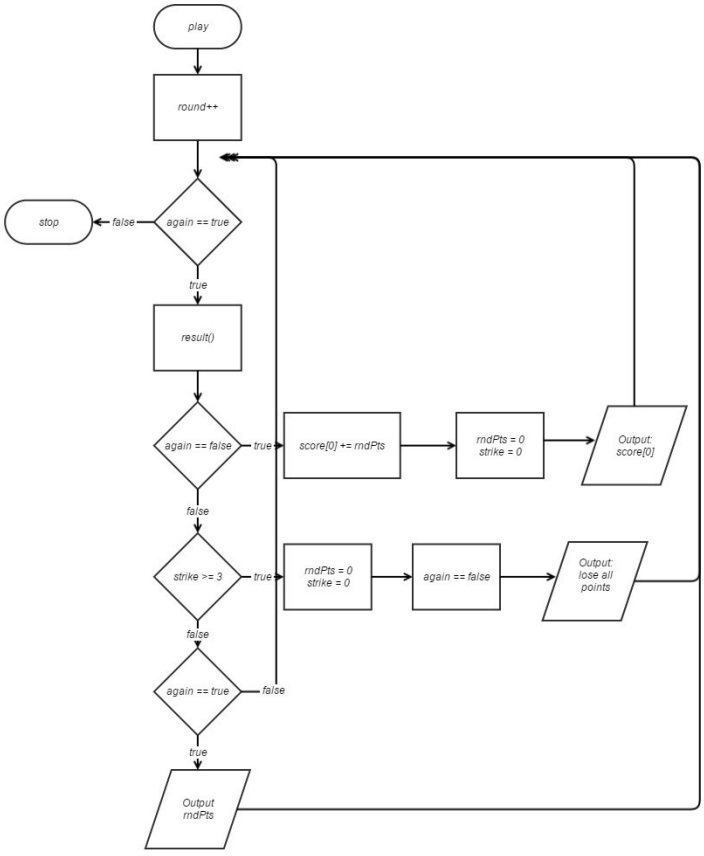

Name
Date
Purpose
System Libraries
iostream
ctime
cstdlib
string
User Libraries
(none)
Global
Constants
none
Function
Prototypes:
header, grab,roll,
clear, play, oplay,
result
A
Main
Declare Variables:
seed, SIZE, score[SIZE]
rndPTs, strike
pTeam, oTeam, pTurn,
round, tRnd, again
header()
Input:
pTeam
oTeam
i++
i > tRnd
false
true
oplay
pTurn = true
pturn = false
play
score[0]
>score[1]
false
score[0]
>score[1]
false
Output:
Tie
true
true
Output:
You win
Output
You lose
return 0
header
Output:
Title Screen
stop


roll(color)
Declare and initialize: val, rslt
Generate random number: val = rand % 6 + 1
color = r
true
val <= 3
false
val == 6
true
false
false
true
false
color = g
true
val == 1
false
val >= 4
true
false
true
false
color = y
true
val <= 2
false
val >= 5
true
rslt = F
rslt= L
rslt = X
true
return rslt


grab
Declare variables: color, val
Generate random number: val = rand % 3 + 1
val == 1
true
color = 'g'
false
val == 2
true
color = 'r'
return color
false
color = 'y'







-Variables Used-

Type	Variable	Description	Location
int	seed	Seed number from time	main
int	SIZE	Size of the array	main
int	score[]	Array to hold the scores	main
unsigned short	rndPts	Points earned this round	main
unsigned short	strike	Number of strikes this round	main
string	pTeam	Name of the player's team	main
string	oTeam	Name of the computer's team	main
bool	pTurn	Is it the player's turn	main
short	round	The round number	main
int	tRnd	Total number of rounds in a game	main
bool	again	Does the user choose to play	main
char	color	Represents the color of the dice	grab
short	val	Random number used to determine the color of the dice	grab
char	val	Random number to determine to outcome	roll
char	rslt	Outcome determined by random number and color of the dice.	roll
char	color	Copied value from grab()	result
char	rslt	Copied value from roll	result
char	rolling	Does the player wish to roll the dice he grabbed	result

-Topics Covered-

bold = used
* = not used

Primitive Data Types:
bool
char
short
int
*float

System Level Libraries:
iostream
*iomanip
cstdlib
ctime
string

Operators
+, -, /, *, %, =
&&, ||
<=, >=, ==. !=
++, -- , +=, -=

Conditions:
if
else if
else
while
*do while
for

Menu:
*switch

```
/*
 * File:   main.cpp
 * Author: Norman Lee
```

```
 * Created on July 17, 2015, 3:09 AM
 * Purpose: Heist Dice - a game based off Zombie Dice.
 */

//System Libraries
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <string>
using namespace std;

//Global Constants

//Function Prototypes
void header();
char grab();
char roll(char);
void clear();
void play(bool &, bool &, unsigned short &, unsigned short &, int [], short &,
string);
void oplay(bool &, bool &, unsigned short &, unsigned short &, int [], short &,
string);
void result(unsigned short &, unsigned short &, bool &, bool &);

//Execution begins here
int main(int argc, char** argv) {
    //Get seed off time and get "random" numbers
    int seed = time(0);
    srand(seed);
    //Declare and Initialize variables
    const int SIZE = 2;
    int score[SIZE] = {};//Array to keep track of both you and your opponents score
    unsigned short rndPts = 0, strike = 0; //points and strikes accrued this round
    string pTeam, oTeam; //Team Names
    bool pTurn = true; //Is it the player's turn?
    short round = 0; //The round number.  Not the Indiana Jones character.
    int tRnd = 5; // Total rounds to be played
    bool again = true; //Play again?
    //Output Start Page
    header();
    //Get names of the teams
    cout << "Enter the name of your team:";
    cin >> pTeam;
    cout << "Enter the name of your opponents:";
    cin >> oTeam;
    cin.ignore();
    //Start the game and loop for the number of rounds
    for (int i = 1; i <= tRnd; i++)
    {
        //Initialize/Reset conditions for the start of the round
```

```
        again = true;
        //Start your turn
        pTurn = true;
        play(again, pTurn, rndPts, strike, score, round, pTeam);
        //Start opponent's turn
        pTurn = false;
        oplay(again, pTurn, rndPts, strike, score, round, oTeam);
        //End of round phase
        clear();
        cout << "Round " << round << " is over. \n" << pTeam << " has " << score[0]
<< " points.\n" << oTeam << " has " << score[1] << " points.\n";
        cout << "Press enter to continue\n";
        cin.ignore();
    }
    //Determine who won
    if (score[0] > score[1])
    {
        cout << pTeam << " Wins!";
    }
    else if (score [1] > score [0])
    {
        cout << oTeam << " Wins...";
    }
    else
    {
        cout << "It's a draw.";
    }
    return 0;
}

//**********************************grab*****************************************
// Purpose: Simulate grabbing a dice out of the bag and getting a random color
// Outputs: color -> g = green, r = red, y = yellow
char grab()
{
    char color;
    short val;
    val = rand()% 10 + 1;
    //determine color of dice grabbed
    if (val <= 5) //50% chance for green
        {
            color = 'g';
            cout << "A Green Dice has been pulled out of the bag.\n";
        }
    else if (val >= 9) //20% chance for red
        {
            color = 'r';
            cout << "A Red Dice has been pulled out of the bag.\n";
        }
    else
```

```
        {
            color = 'y';//30% chance for yellow
            cout << "A Yellow Dice has been pulled out of the bag.\n";
        }
    return (color);
}

//**********************************roll*****************************************
//Purpose: Roll a dice and get a weighted outcome based on the color
//Input: dice -> the color of dice
//Output: rslt -> weighted results, green has the highest positive outcome, reds
have the lowest
char roll(char dice)
{
    char val, rslt;
    val = rand()%6 + 1;
    if (dice == 'r') //Red dice, weighted with 50% chance of a strike
    {
        if (val <= 3) //%50 chance of a strike
        {
            rslt = 'X'; //you got caught/a strike
        }
        else if (val == 6) //1/6 chance for a point
        {
            rslt = 'L'; //you got loot/a point
        }
        else //1/3 chance to get away
        {
            rslt = 'F'; //you failed to get loot but got away
        }
    }
    else if (dice == 'g') //Green dice, weighted with a 50% chance to get you Loot
    {
        if (val == 1)//1/6 chance of a strike
        {
            rslt = 'X'; //you got caught/a strike
        }
        else if (val >= 4)//1/2 chance of a point
        {
            rslt = 'L'; //you got loot/a point
        }
        else//1/3 chance of getting
        {
            rslt = 'F'; //you failed to get loot but got away
        }
    }
    else if (dice == 'y') //Yellow dice, weighted with even odds for all outcomes
    {
        if (val <= 2)
        {
```

```
            rslt = 'X'; //you got caught/a strike
        }
        else if (val >= 5)
        {
            rslt = 'L'; //you got loot/a point
        }
        else
        {
            rslt = 'F'; //you failed to get loot but got away
        }
    }
    return (rslt);
}

//*************************result**********************************
//Purpose: Add points or strike and output ascii picture
//Inputs:
//    rndPts -> points for this round
//    strike -> number of strikes
//      again  -> play again?
//      pTurn  -> is it the players turn
//Outputs:
//    rndPts
//    strike
void result(unsigned short &rndPts, unsigned short &strike, bool &again, bool
&pTurn)
{
    char rslt, color, rolling; // copied values for color and the value you get
from rolling
    color = grab(); //grab a dice from the bag and find out which color you got
      //Check if user wants to roll
    if (pTurn == true)
    {
        cout << "Do you wish to roll the dice?(y/n)";
        cin >> rolling;
        cin.ignore();
        if (rolling == 'n')
        {
            again = false;
        }
    }
    if (pTurn == false || rolling == 'y')
    {
        rslt = roll(color); //roll the dice and get the value
        clear();
        //Output  representation of the dice and increment strikes or points based
on roll result
        if (rslt == 'L')
        {
            cout << ".-----------.\n"
```

```
                << "|   $$$$$   |\n"
                << "|    $ $    |\n"
                << "|  $     $  |\n"
                << "| $       $ |\n"
                << "|  $$$$$$$  |\n"
                << "'-----------'\n";
            cout << "Successfully stole some loot!\n";
            rndPts++;
        }
        else if (rslt == 'X')
        {
            cout << ".-----------.\n"
                << "| x       x |\n"
                << "|   x   x   |\n"
                << "|     x     |\n"
                << "|   x   x   |\n"
                << "| x       x |\n"
                << "'-----------'\n";
            cout << "Got caught and a strike was placed on your record.\n";
            strike++;
        }
        else
        {
            cout << ".-----------.\n"
                << "| RRRRR     |\n"
                << "| R   R     |\n"
                << "| R   RRRR  |\n"
                << "| R       R |\n"
                << "| RRRRRRRRR |\n"
                << "'-----------'\n";
            cout << "Failed at the attempt but got away.\n";
        }
    }
}
//Purpose: Generate a header with information
void header()
{
    cout << " . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. ..-----------. . . . . .\n"
         << ". . . . . $ . .$. .$$$$ . $ .$$$$$. $$$$$$. . . 8888. 8 .888. .
8888 . . . |   $$$$$   |. . . . . \n"
         << " . . . . .$. . $ . $ . . .$.$  . .$. .$. . . . .8. .8.8.8. .8. 8 . . .
. .|    $ $    | . . . . .\n"
         << ". . . . . $$$$$$. .$$$. . $ .$$$$ . . $ . . . . 8 . 8 8 8 . . .
888. . . . |  $     $  |. . . . . \n"
         << " . . . . .$. . $ . $ . . .$.$. . $$. .$. . . . .8. .8.8.8. .8. 8 . . .
. .| $       $ | . . . . .\n"
         << ". . . . . $ . .$. .$$$$ . $ .$$$$$. . $ . . . . 8888. 8 .888. .
8888 . . . |  $$$$$$$  |. . . . . \n"
         << " . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

```
. .'----------' . . . . .\n"
         << "How to play Heist Dice: Every round you get to roll the dice.\n\n"
         << "There are 3 different outcomes you get from rolling: A loot bag,
shoes, and a strike.\n"
         <<
"-------------------------------------------------------------------------------
---------------\n"
         << "Loot bags: you score a point for the round.\n"
         << "Shoes: nothing good or bad happens.\n"
         << "Strikes: If you get 3 strikes, the round ends and you lose any points
you made for that round.\n"
         << ".-----------.     .-----------.     .-----------.\n"
         << "|   $$$$$   |     | RRRRR     |     | x       x |\n"
         << "|    $ $    |     | R   R     |     |   x   x   |\n"
         << "|  $     $  |     | R   RRRR  |     |     x     |\n"
         << "| $       $ |     | R       R |     |   x   x   |\n"
         << "|  $$$$$$$  |     | RRRRRRRRR |     | x       x |\n"
         << "'-----------'     '-----------'     '-----------'\n\n"
         << "There are 3 types of die: Green, Red and Yellow.\n"
         <<
"-------------------------------------------------------------------------------
---------------\n"
         << "Green: Has the highest amount of loot bags and only 1 strike.\n"
         << "Red: Has the highest amount of strikes and only 1 loot bag.\n"
         << "Yellow: All outcomes have an equal chance.\n"
         <<
"-------------------------------------------------------------------------------
---------------\n"
         << "After you grab a dice from the bag you can choose to stop and keep the
points you have accrued.\n"
         << "There are 5 rounds, you and your opponent take turns rolling for each
round.\n";
    cout << "<Press Enter to continue>\n";
    cin.ignore();
}

//Purpose: uses 40 new lines to clear the screen
void clear()
{
    for(int i = 1; i <=40; i++)
    {
    cout << endl;
    }
}
//*******************************play*********************
//Purpose: Player's turn to play the game
//Inputs:
// again -> play again, true or false
// rndPts -> points this round
// strike -> strikes this round
```

```
// score[] -> total scored points
// round -> the round number
// oTeam -> the computer's team name
//Outputs:
// again -> play again?
// rndPts -> points this round
// strike -> strikes this round
// score[] -> total scored points
// oTeam -> the computer's team name
void play(bool &again, bool &pTurn, unsigned short &rndPts, unsigned short &strike,
int score[], short &round, string pTeam)
{
    round++; //Add one to start the round
    while(again == true)
    {
        cout << "Round " << round << "<Your turn>" << endl;
        result(rndPts, strike, again, pTurn);
        if (again == false)
        {
            score[0] += rndPts;
            rndPts = 0;
            strike = 0;
            cout << "Added point(s) to total. " << pTeam << " has " << score[0] <<
" total points.\n\n";
        }
        if (strike >= 3)
        {
            cout << pTeam << " member <" << round <<"> has 3 strikes. Facing life
in prison, you give up all your loot for a plea deal.\n\n";
            rndPts = 0;
            again = false;
            strike = 0;
        }
        else
        {
            if (again == true)
            {
            cout << pTeam << " member <" << round << "> has " << rndPts << " points
this round and " << strike << " strikes. " << pTeam << " has " << score[0] << "
total points.\n\n";
            }
        }
        cout << "<Press Enter to continue>\n";
        cin.ignore();
    }
}
//************************************oplay*************************************
*
//Purpose: Have the computer opponent play
//Inputs:
```

```
// again -> play again, true or false
// rndPts -> points this round
// strike -> strikes this round
// score[] -> total scored points
// round -> the round number
// oTeam -> the computer's team name
//Outputs:
// again -> play again?
// rndPts -> points this round
// strike -> strikes this round
// score[] -> total scored points
// oTeam -> the computer's team name
void oplay(bool &again, bool &pTurn, unsigned short &rndPts, unsigned short
&strike, int score[], short &round, string oTeam)
{
    again = true;
    while (again == true)
    {
        result(rndPts, strike, again, pTurn);
        cout << oTeam << " member " << round <<" has "<< rndPts << " points and "
<< strike << " strikes.\n" << endl;
        cout << "Round " << round << " <Opponent's turn>" << endl;
        cout << "<Press enter to continue>\n";
        cin.ignore();
        if (strike >= 2)
        {
            //Add round points to total score and reinitialize
            score[1] += rndPts;
            rndPts = 0;
            strike = 0;
            cout << "Added point(s) to total. " << score[1] << " total points for
"<< oTeam <<" .\n";
            again = false;
            cout << "<Press enter to continue>\n";
            cin.ignore();
        }
    }
}
```