YAHTZEE

A DICE GAME

Ireoluwa Dairo

CSC/CIS 5

45744

Summer 2024

INTRODUCTION

This documentation provides an overview of a Yahtzee game implemented in C++. It covers the game's mechanics and programming approach.

HOW THE GAME WORKS

Yahtzee is a dice game where players roll five dice and attempt to achieve specific combinations to score points. The game involves rolling dice, holding some dice while re-rolling others, and calculating scores based on various combinations.

OBJECT OF THE GAME

The objective of the game is to accumulate the highest possible score by rolling five dice and achieving various scoring combinations over a series of rounds.

RULES OF THE GAME

1. Roll five dice and attempt to achieve the best combinations for points.
2. You can roll the dice up to three times per turn.
3. Specific combinations like Yahtzee, Large Straight, and Four of a Kind have different point values.

GAMEPLAY MECHANICS

- Rolling Dice: The player rolls dice that are not held.
- Holding Dice: The player can choose to hold specific dice between rolls.
- Scoring: Points are awarded based on the combinations rolled.

SIMILARITIES TO THE ORIGINAL GAME

- Dice rolling and holding mechanics are similar.
- Scoring combinations and their point values match the traditional Yahtzee game.

DIFFERENCES FROM THE ORIGINAL GAME

- This implementation does not include all Yahtzee scoring features.
- The game does not support multiplayer.

MY APPROACH TO THE GAME (PROJECT 1 TO 2)

I started editing the code following the checklist and used the textbook as a guide. I began by changing the roll mechanism to use arrays, then split the code into function prototypes. I initially used two display score prototypes—one to display the scores after each round and another to display the scores after the final round. I later changed this to use overloading instead.

For username input validation, I used default arguments so that if the user does not enter a username, it automatically defaults to "player." I then changed the rolls left to a static variable. I modified the logic of the quit case to use exit(0) rather than setting the round number to the final round. I also created a new function prototype to display the final results, similar to the one that displays scores.

To properly check the special combinations, I added a bubble sort to sort the dice values before checking for special combinations. I initially had a problem with the special combination logic where a four of a kind would also display a three of a kind. I fixed this by changing the logic from an if statement to an if-else if statement and worked my way backwards from Yahtzee down to three of a kind.

To use 2-dimensional arrays, I decided to create another version of the same code. I changed the single-dimensional array for the dice and hold to a singular diceInfo 2D array. I created a new version of the code because I wasn't sure how I was going to implement the selection sort. In this version, I sorted all the points into descending order using selection sort and then used a linear search to find the highest and lowest points. This helped me determine the round in which I got each point.

However, I ran into an issue. After using selection sort to organize the points, the linear search would take the sorted order as the order of the rounds. For example, if I had 5 rounds and scored 2 in round 1, 10 in round 2, 6 in round 3, 4 in round 4, and 50 in round 5, after sorting, the linear search would take 50 as round 1 and 2 as round 5. To prevent this mix-up, I decided to use the linear search before sorting with selection sort. This way, the rounds wouldn't get mixed up, and the original order would be preserved.

I then ran another round of tests and realized that if I held a die in a round, that die would be held for the rest of the rounds and would not generate a new number. So, I changed the logic of my game loop to prevent this from happening.

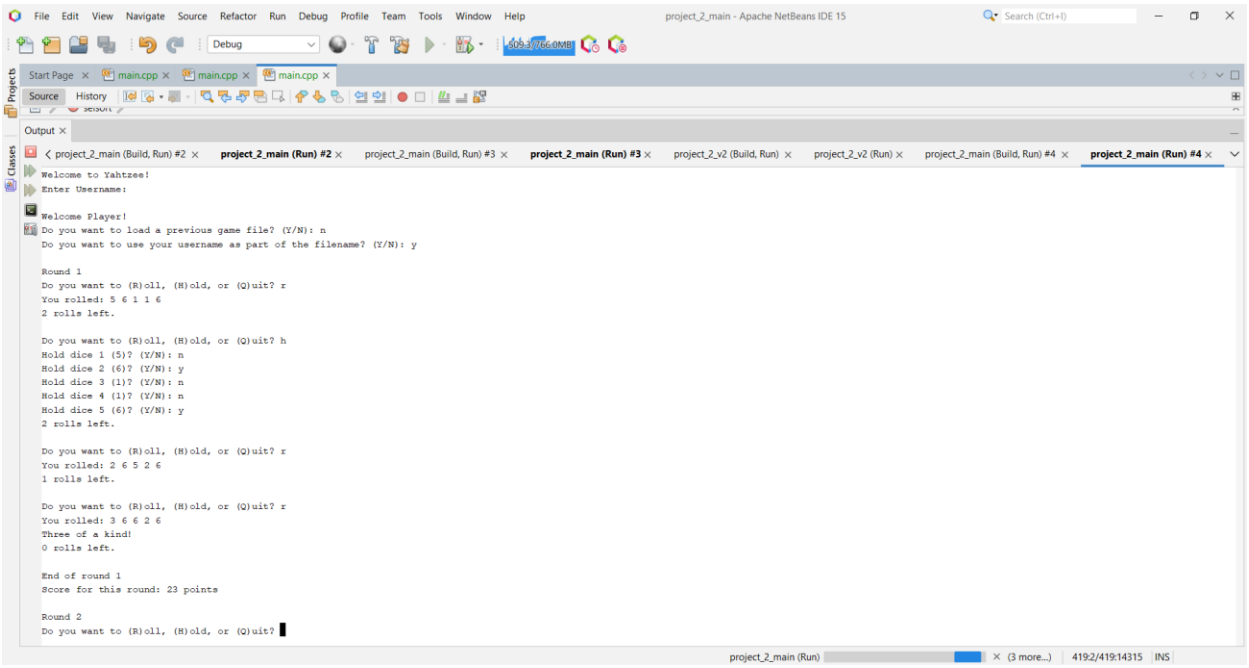
THE LOGIC OF IT ALL (PSEUDOCODE)

```
START Program

 // System Libraries
 INCLUDE libraries (iostream, ctime, cstdlib, fstream, iomanip, string, cmath)

 // Function Prototypes
 DECLARE functions

 FUNCTION main
  INITIALIZE random seed
  DECLARE uname, fname, out, scores[13] = {0}

  CALL wlcmMsg()
  IF ldPvGme(fname) returns false THEN
   CALL dFilnme(uname, fname)
  ENDIF
  CALL savGame(out, uname, fname)
  CALL Gmeloop(scores, out)
  CALL dspscre(scores, out)
  CLOSE out file
  RETURN 0
 END FUNCTION main

 FUNCTION Gmeloop(scores, out)
  FOR round 0 to 12
   INITIALIZE dice[5], hold[5] = {false}, rllsLft = 3
   PRINT "Round {round + 1}"
   WRITE "Round {round + 1}" to out

   DO
    GET user choice Gchce (R)oll, (H)old, or (Q)uit
    SWITCH Gchce
     CASE 'R' or 'r': CALL rllDice(dice, hold), CALL dspRoll(dice, out), CALL Spclcmb(dice, out), rllsLft--
     CASE 'H' or 'h': CALL hldDice(dice, hold)
     CASE 'Q' or 'q': CALL exitGme(sum of scores), EXIT program
     DEFAULT: PRINT "Invalid choice!"
    END SWITCH
    IF rllsLft < 3 THEN PRINT "{rllsLft} rolls left."
   WHILE rllsLft > 0 AND Gchce NOT 'Q' AND Gchce NOT 'q'

   scores[round] = sum of dice
   CALL dspscre(round + 1, scores[round], out)
  END FOR
 END FUNCTION Gmeloop

 FUNCTION wlcmMsg(uname = "Player")
  GET user input for username
  IF input is not empty THEN uname = input
  PRINT "Welcome {uname}!"
 END FUNCTION wlcmMsg
```

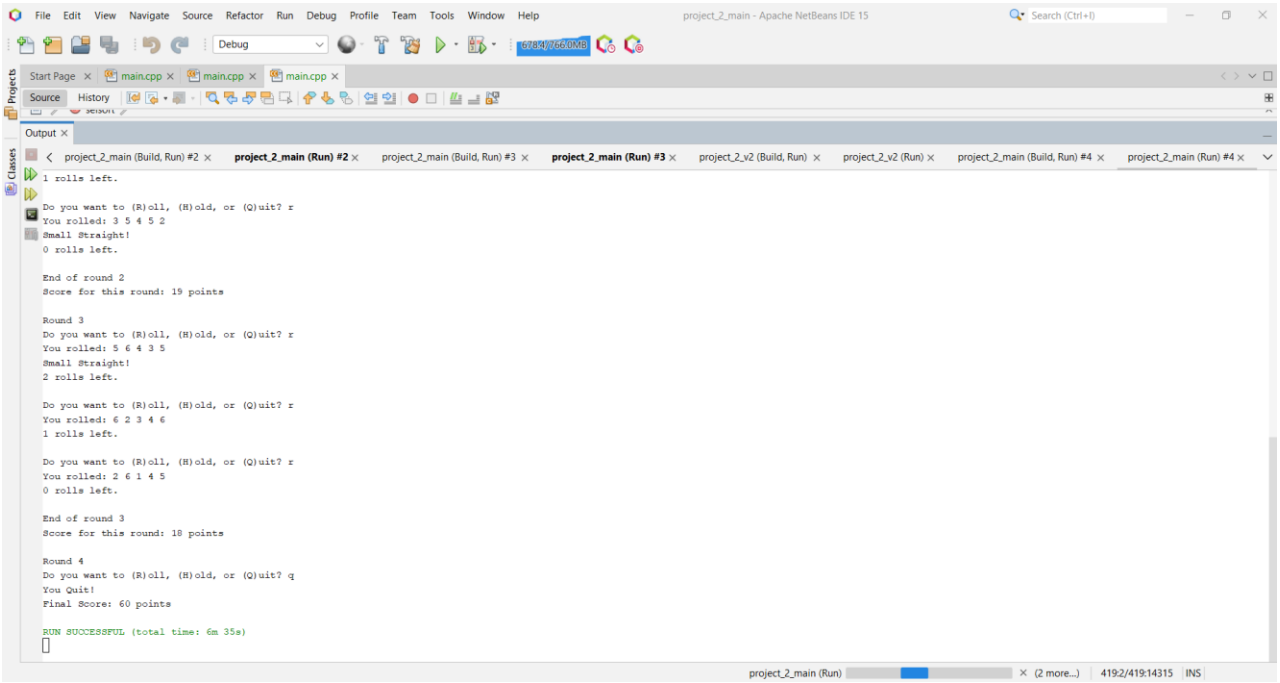
```
FUNCTION ldPvGme(fname)
  GET user input to load previous game (Y/N)
  IF 'Y' THEN
    GET filename from user
    OPEN file fname for reading
    IF file is open THEN
      READ and PRINT lines from file
      CLOSE file
      RETURN true
    ELSE PRINT "File not found."
  ENDIF
  RETURN false
END FUNCTION ldPvGme

FUNCTION dFilnme(uname, fname)
  GET user input to use username as part of filename (Y/N)
  IF 'Y' THEN fname = "{uname} game results.txt"
  ELSE GET filename from user
END FUNCTION dFilnme

FUNCTION savGame(out, uname, fname)
  OPEN out file fname for writing
  IF file is open THEN
    WRITE "Welcome to Yahtzee!", "Player: {uname}" to out
END FUNCTION savGame

FUNCTION rllDice(dice, hold)
  FOR each element in dice
    IF hold[element] is false THEN dice[element] = random number 1-6
END FUNCTION rllDice

FUNCTION dspRoll(dice, out)
  PRINT and WRITE "You rolled: ", elements of dice
END FUNCTION dspRoll

FUNCTION Spclcmb(dice, out)
  SORT dice
  IF Yahtzee(dice) THEN PRINT and WRITE "Yahtzee!"
  ELSE IF FrOAKnd(dice) THEN PRINT and WRITE "Four of a kind!"
  ELSE IF FullHse(dice) THEN PRINT and WRITE "Full House!"
  ELSE IF LrgStht(dice) THEN PRINT and WRITE "Large Straight!"
  ELSE IF SmlStht(dice) THEN PRINT and WRITE "Small Straight!"
  ELSE IF ThOAKnd(dice) THEN PRINT and WRITE "Three of a kind!"
END FUNCTION Spclcmb

FUNCTION hldDice(dice, hold)
  FOR each element in dice
    GET user input to hold dice[element] (Y/N)
    SET hold[element] to true if 'Y'
END FUNCTION hldDice

FUNCTION exitGme(ttlScre)
  PRINT "Total score is {ttlScre}", "Thank you for playing!"
```

```
  EXIT program
END FUNCTION exitGme

FUNCTION dspscre(round, rndScre, out)
  PRINT and WRITE "Round {round} score: {rndScre}"
END FUNCTION dspscre

FUNCTION dspscre(scores, out)
  DECLARE ttlScre = sum of scores
  PRINT and WRITE "Total score is {ttlScre}"
END FUNCTION dspscre

FUNCTION bblsort(array, size)
  SORT array using bubble sort
END FUNCTION bblsort

FUNCTION selsort(array, size)
  SORT array using selection sort
END FUNCTION selsort

FUNCTION linsrch(scores, hghst, lwst, hRnd, lRnd)
  FIND highest and lowest scores and their rounds
END FUNCTION linsrch

FUNCTION Yahtzee(dice)
  RETURN true if all elements in dice are equal
END FUNCTION Yahtzee

FUNCTION FrOAKnd(dice)
  RETURN true if 4 or more elements in dice are equal
END FUNCTION FrOAKnd

FUNCTION FullHse(dice)
  RETURN true if dice has a 3-of-a-kind and a pair
END FUNCTION FullHse

FUNCTION LrgStht(dice)
  RETURN true if dice has 5 consecutive numbers
END FUNCTION LrgStht

FUNCTION SmlStht(dice)
  RETURN true if dice has 4 consecutive numbers
END FUNCTION SmlStht

FUNCTION ThOAKnd(dice)
  RETURN true if 3 or more elements in dice are equal
END FUNCTION ThOAKnd

FUNCTION smArray(array, size)
  RETURN sum of elements in array
END FUNCTION smArray

END Program
```

PROOF OF WORKING CODE



```
Welcome to Yahtzee!
Enter Username:

Welcome Player!
Do you want to load a previous game file? (Y/N): n
Do you want to use your username as part of the filename? (Y/N): y

Round 1
Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 5 6 1 1 6
2 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? h
Hold dice 1 (5)? (Y/N): n
Hold dice 2 (6)? (Y/N): y
Hold dice 3 (1)? (Y/N): n
Hold dice 4 (1)? (Y/N): n
Hold dice 5 (6)? (Y/N): y
2 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 2 6 5 2 6
1 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 3 6 6 2 6
Three of a kind!
0 rolls left.

End of round 1
Score for this round: 23 points

Round 2
Do you want to (R)oll, (H)old, or (Q)uit?
```



```
1 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 3 5 4 5 2
Small Straight!
0 rolls left.

End of round 2
Score for this round: 19 points

Round 3
Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 5 6 4 3 5
Small Straight!
2 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 6 2 3 4 6
1 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 2 6 1 4 5
0 rolls left.

End of round 3
Score for this round: 18 points

Round 4
Do you want to (R)oll, (H)old, or (Q)uit? q
You Quit!
Final Score: 60 points

RUN SUCCESSFUL (total time: 6m 35s)
```

```
Welcome to Yahtzee!
Enter Username: ireoluwa

Welcome ireoluwa!
Do you want to load a previous game file? (Y/N): y
Enter the filename to load the game results: ee game results.txt
Welcome to Yahtzee!
Player: ee

Round 1
Rolled: 2 1 4 2 4
Rolled: 2 6 2 5 3
Rolled: 4 6 5 1 3
Small Straight!
Score for this round: 19 points
Total score: 19 points

Round 2
Rolled: 5 3 2 1 4
Large Straight!
Rolled: 6 5 6 6 5
Full House!
Rolled: 5 3 2 2 6
Score for this round: 18 points
Total score: 37 points

Round 3
Rolled: 6 6 1 6 4
Three of a kind!
Rolled: 1 1 3 6 2
Rolled: 5 2 5 4 3
Small Straight!
Score for this round: 19 points
Total score: 56 points
```

```
Three of a kind!
1 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 4 6 2 4 2
0 rolls left.

End of round 12
Score for this round: 18 points

Round 13
Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 3 6 6 3 2
2 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 4 6 6 3 4
1 rolls left.

Do you want to (R)oll, (H)old, or (Q)uit? r
You rolled: 3 2 5 6 5
0 rolls left.

End of round 13
Score for this round: 21 points

Final score: 256
Average points: 20 point(s) per round
Highest score: 25 points in round 8
Lowest score: 15 points in round 3
Scores in descending order: 25 22 21 21 21 20 20 19 18 18 18 18 15

RUN SUCCESSFUL (total time: 3m 29s)
```

THE CODE

```
/*
 * Author: Ireoluwa
 * Created on July 21, 4:18 pm
 * Purpose: a game of Yahtzee
 */

// System Libraries
#include <iostream>   // I/O library for input and output operations
#include <ctime>      // Library to work with time functions
#include <cstdlib>    // Random number generation
#include <fstream>    // File input and output
#include <iomanip>    // I/O manipulator
#include <string>     // string library
#include <cmath>      // Math library
using namespace std;

//User Libraries

//Global Constants - Mathematical, Scientific, Conversions

//Higher Dimensions go here. No Variables

//Function Prototypes
void wlcmMsg (string uname = "Player");                // welcome Message
void bblsort (int[], int);                             // bubble sort Array
void dFilnme (string&, string&);                       // determine Filename
void savGame (fstream&, const string&, const string&);   // Saved Game
void rllDice (int[], bool[]);                          // roll Dice
void dspRoll (const int[], fstream&);                  // display Roll
void Spclcmb (const int[], fstream&);                  // Special Combinations
void hldDice (int[], bool[]);                          // hold Dice
void dspscre (int, int, fstream&);                     // display Round Result
void dspscre (const int[], fstream&);                  // display Final Result
void exitGme (int);                                    // exit Game
void Gmeloop (int[], fstream& out);                    // Game loop
void linsrch (const int[], int&, int&, int&, int&);      // linear search
void selsort (int[], int);                             // selection sort
bool Yahtzee (const int[]);                            // Yahtzee combination
bool FrOAKnd (const int[]);                            // Four Of A Kind
bool FullHse (const int[]);                            // Full House
bool LrgStht (const int[]);                            // Large Straight
bool SmlStht (const int[]);                            // Small Straight
bool ThOAKnd (const int[]);                            // Three Of A Kind
bool ldPvGme (const string&);                          // load Previous Game
int  smArray (const int[], int);                       // sum Array

//Execution Begins here
int main(int argc, char *argv[]) {
    // Setting the random number seed
    srand(static_cast<unsigned int>(time(0)));

    // Declaring Variables
```

```
    string uname, fname;
    fstream out;

    // Initialize Variables
    int scores[13] = {0};

    // Welcome message and prompt for username
    wlcmMsg();

    //Gamefile choice
    if (!ldPvGme(fname)) {dFilnme(uname, fname);}
    savGame(out, uname, fname);

    // Game loop to manage multiple rounds
    Gmeloop(scores, out);
    dspscre(scores, out);
    out.close();
    return 0;}

// Function Definitions
void Gmeloop(int scores[], fstream& out) {          // loop to manage all rounds
    for (int round = 0; round < 13; round++) {      //13 rounds
        int dice[5];                        // Initialize dice and hold arrays
        bool hold[5];

        // Reset the hold array for each round
        for (int i = 0; i < 5; i++) {
            hold[i] = false;
        }
        static int rllsLft = 3;                     // Initialize rolls left
        cout << "Round " << round + 1 << endl;      // Display round number
        out << "Round " << round + 1 << endl;

        char Gchce;                         // Get user input for each round
        do {
            cout << "Do you want to (R)oll, (H)old, or (Q)uit? ";
            cin >> Gchce;                           // Prompt user for action
            cin.ignore();

            switch (Gchce) {                        // Handle user input
                case 'R':                           // Roll dice
                case 'r':
                    rllDice(dice, hold);
                    dspRoll(dice, out);
                    Spclcmb(dice, out);
                    rllsLft--;          // Decrement rolls left
                    break;

                case 'H':                               // hold dice
                case 'h':
                    hldDice(dice, hold);
                    break;

                case 'Q':                               //quit
```

```
            case 'q':
                exitGme(smArray(scores, round));
                exit(0);

            default:                // If the user enters an invalid choice
                cout << "Invalid choice!" << endl;
        }
        // Display remaining rolls
        if (rllsLft < 3){cout << rllsLft << " rolls left." << endl<<endl;}
    } while (rllsLft > 0 && Gchce != 'Q' && Gchce != 'q');
        rllsLft = 3;        // Reset rolls left for next round

    scores[round] = smArray(dice, 5);     // Calculate score each round
    dspscre(round + 1, scores[round], out);     // Display round result
  }
}

void wlcmMsg(string uname) {            //welcome messasge
  string input;
  cout << "Welcome to Yahtzee!" << endl;
  cout << "Enter Username: ";         //prompt user for username
  getline(cin, input);
  if (!input.empty()) { uname = input;}
  cout << endl << "Welcome " << uname << "!" << endl;}

bool ldPvGme(const string& fname) {         //gamefile choice
  char lodChce;
  cout << "Do you want to load a previous game file? (Y/N): ";
  cin >> lodChce;
  cin.ignore();
  while (lodChce != 'Y' && lodChce != 'y' && lodChce != 'N' && lodChce != 'n')
  {cout << "Invalid Input. Input 'Y' or 'N': ";           //input validation
  cin >> lodChce;
  cin.ignore();}

  if (lodChce == 'Y' || lodChce == 'y') {
    string fname;
    cout << "Enter the filename to load the game results: ";
    getline(cin, fname);                //prompt user for game file name

    ifstream in(fname);
    if (in.is_open()) {
      string line;
      while (getline(in, line)) {
        cout << line << endl;
      }
      cout << "Try to beat your previous score" << endl;
      cout << "Starting New Game!" << endl << endl;
      in.close();
      return true;
    } else {                //error message
      cout << "File not found. Starting a new game." << endl << endl;
    }
  }
```

```
    return false;
}

void dFilnme(string& uname, string& fname) {            //determine file name
    char svChce;
    cout << "Do you want to use your username as part of the filename? (Y/N): ";
    cin >> svChce;
    cin.ignore();
    while (svChce != 'Y' && svChce != 'y' && svChce != 'N' && svChce != 'n') {
    cout << "Invalid Input. Input 'Y' or 'N': ";        //input validation
    cin >> svChce;
    cin.ignore();}

    if (svChce == 'Y' || svChce == 'y') {
        fname = uname + " game results.txt";
    } else {
        cout << "Enter the filename to save the game results: ";
        getline(cin, fname);
        while (fname.empty()) {             //input validation
        cout << "Filename cannot be empty. Please enter a filename: ";
        getline(cin, fname);}
    }cout << endl;
    return;
}

void savGame(fstream& out, const string& uname, const string& fname) {
    out.open(fname, ios::out);
    if (out.is_open()) {
        out << "Welcome to Yahtzee!" << endl;       //begin game
        out << "Player: " << uname << endl << endl;
    }
}

void rllDice(int dice[], bool hold[]) {         //roll dice function
    for (int i = 0; i < 5; i++) {
        if (!hold[i]) {
            dice[i] = rand() % 6 + 1;
        }
    }
}

void dspRoll(const int dice[], fstream& out) {      //display rolled dice
    cout << "You rolled: ";
    out << "Rolled: ";
    for (int i = 0; i < 5; i++) {
        cout << dice[i] << " ";
        out << dice[i] << " ";
    }
    cout << endl;
    out << endl;
}

void Spclcmb(const int dice[], fstream& out) {      //determine combinations
    int srtdDce[5];
```

```
    for (int i = 0; i < 5; i++) {
        srtdDce[i] = dice[i];
    }
    bblsort(srtdDce, 5);

    if (Yahtzee(srtdDce)) {             //yahtzee combination
        cout << "Yahtzee!" << endl;
        out << "Yahtzee!" << endl;
    }
    else if (FrOAKnd(srtdDce)) {        //four of a kind
        cout << "Four of a kind!" << endl;
        out << "Four of a kind!" << endl;
    }
    else if (FullHse(srtdDce)) {        //full house
        cout << "Full House!" << endl;
        out << "Full House!" << endl;
    }
    else if (LrgStht(srtdDce)) {        //large straight
        cout << "Large Straight!" << endl;
        out << "Large Straight!" << endl;
    }
    else if (SmlStht(srtdDce)) {        //small straight
        cout << "Small Straight!" << endl;
        out << "Small Straight!" << endl;
    }
    else if (ThOAKnd(srtdDce)) {            //three of a kind
        cout << "Three of a kind!" << endl;
        out << "Three of a kind!" << endl;
    }
}

void hldDice(int dice[], bool hold[]) {             //hold dice
    for (int i = 0; i < 5; i++) {
        char choice;
        cout << "Hold dice " << i + 1 << " (" << dice[i] << ")? (Y/N): ";
        cin >> choice;
        hold[i] = (choice == 'Y' || choice == 'y');
        while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n')
        {cout << "Invalid Input. Input 'Y' or 'N': ";
        cin >> choice;
        cin.ignore();}
    }
}

void dspscre(int round, int rndScre, fstream& out) {      // display round score
    cout << "End of round " << round << endl;
    cout << "Score for this round: " << rndScre << " points" << endl << endl;
    out << "Score for this round: " << rndScre << " points" << endl << endl;
}

void dspscre(const int scores[], fstream& out) {          // display final score
    int ttlScre = smArray(scores, 13);
    float aScore = ttlScre / 13.0f;
    int rndAvrg = round(aScore);
```

```
    int hghst, lwst, hRnd, lRnd;
    linsrch(scores, hghst, lwst, hRnd, lRnd);

    cout << "Final score: " << setw(3) << setfill('0') << ttlScre << endl
         << "Average points: " << rndAvrg << " point(s) per round" << endl
         << "Highest score: " << hghst << " points in round " << hRnd << endl
         << "Lowest score: " << lwst << " points in round " << lRnd << endl;

    out  << "Final score: " << setw(3) << setfill('0') << ttlScre << endl
         << "Average points: " << rndAvrg << " point(s) per round" << endl
         << "Highest score: " << hghst << " points in round " << hRnd << endl
         << "Lowest score: " << lwst << " points in round " << lRnd << endl;

    // Sort and display scores in descending order
    int stdScrs[13];
    for (int i = 0; i < 13; i++) {
        stdScrs[i] = scores[i];
    }
    selsort(stdScrs, 13);

    cout << "Scores in descending order: ";
    out << "Scores in descending order: ";
    for (int i = 0; i < 13; i++) {
        cout << stdScrs[i] << " ";
        out << stdScrs[i] << " ";
    }
    cout << endl;
    out << endl;
}

void exitGme(int ttlScre) {                 //exit game
    cout << "You Quit!" << endl;
    cout << "Final Score: " << ttlScre << " points" << endl;
}

int smArray(const int arr[], int size) {        //sum scores
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

bool Yahtzee(const int dice[]) {
    bool form = false;
    // Check if dice form a Yahtzee
    for (int i = 1; i < 5; i++) {
        if (dice[i] == dice[0]) {
            form = true;
        } else {
            form = false;
            break;
        }
```

```
    }
    return form;
}

bool FrOAKnd(const int dice[]) {        //check for four of a kind
    if ((dice[0] == dice[1] && dice[1] == dice[2] && dice[2] == dice[3]) ||
        (dice[1] == dice[2] && dice[2] == dice[3] && dice[3] == dice[4])) {
        return true;
    }
    return false;
}

bool FullHse(const int dice[]) {                    //check for full house
    if ((dice[0] == dice[1] && dice[1] == dice[2] && dice[3] == dice[4]) ||
        (dice[0] == dice[1] && dice[2] == dice[3] && dice[3] == dice[4])) {
        return true;
    }
    return false;
}

bool LrgStht(const int dice[]) {            //check for large straight
    if ((dice[0] == 1 && dice[1] == 2 && dice[2] == 3 && dice[3] == 4 && dice[4]
         == 5) || (dice[0] == 2 && dice[1] == 3 && dice[2] == 4 && dice[3] == 5
         && dice[4] == 6)) {
        return true;
    }
    return false;
}

bool SmlStht(const int dice[]) {        //check for small straight
    int counts[6] = {0};
    for (int i = 0; i < 5; i++) {
        counts[dice[i] - 1]++;
    }
    for (int i = 0; i < 3; i++) {
        if (counts[i] > 0 && counts[i+1] > 0 && counts[i+2] > 0 &&
                counts[i+3] > 0) {
            return true;
        }
    }
    return false;
}

bool ThOAKnd(const int dice[]) {        //check for three of a kind
    for (int i = 0; i < 3; i++) {
        if (dice[i] == dice[i+1] && dice[i] == dice[i+2]) {
            return true;
        }
    }
    return false;
}

void bblsort(int arr[], int size) {             //bubble sort
    for (int i = 0; i < size - 1; i++) {
```

```
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void linsrch(const int scrs[], int& hghst, int& lwst, int& hRnd, int& lRnd) {
    hghst = scrs[0];                    //linear search
    lwst = scrs[0];
    hRnd = 1;
    lRnd = 1;

    for (int i = 1; i < 13; i++) {
        if (scrs[i] > hghst) {
            hghst = scrs[i];
            hRnd = i + 1;
        }
        if (scrs[i] < lwst) {
            lwst = scrs[i];
            lRnd = i + 1;
        }
    }
}

void selsort(int arr[], int size) {             //selection sort
    for (int i = 0; i < size - 1; i++) {
        int maxIndx = i;
        for (int j = i + 1; j < size; j++) {
            if (arr[j] > arr[maxIndx]) {
                maxIndx = j;
            }
        }
        if (maxIndx != i) {
            int temp = arr[i];
            arr[i] = arr[maxIndx];
            arr[maxIndx] = temp;
        }
    }
}
```

Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
6		Functions			
	3	Function Prototypes	24-45	4	Always use prototypes
	5	Pass by Value	371	4	
	8	return	306	4	A value from a function
	9	returning boolean	151,156,326,328,334,336 343,345,356,359,365,368	4	
	10	Global Variables	None	XXX	Do not use global variables -100 pts
	11	static variables	77	4	
	12	defaulted arguments	24	4	
	13	pass by reference	159,181	4	
	14	overloading	32, 33	5	
	15	exit() function	104	4	
7		Arrays			
	1 to 6	Single Dimensioned Arrays	75,76	3	
	7	Parallel Arrays		2	
	8	Single Dimensioned as Function Arguments	57	2	
	9	2 Dimensioned Arrays	75 version 3	2	Emulate style in book/in class repositiory
	12	STL Vectors		2	
		Passing Arrays to and from Functions	67,95, 96, 97, 108,118 throughout	5	
		Passing Vectors to and from Functions		5	
8		Searching and Sorting Arrays			
	3	Bubble Sort	375	4	
	3	Selection Sort	405	4	
	1	Linear or Binary Search	387	4	
****** Not required to show			Total	70	Other 30 points from Proj 1 first sheet tab