Project 2

Title: Yahtzee

Course: CIS 5

Section: 40652

Name: Allison Ohara

Date: 02/0818

Table of Contents

Introduction:

Title: Yahtzee

My project is the game of Yahtzee. This program allows one or two players to play the game of Yahtzee. If one player is playing, then he wins if he gets more than 225 points. If two players play, then whoever scores more points wins. If one player is playing, then whoever gets more points wins.

Rules of the Game:

The goal of the game is to get as many points as possible.

First, the file containing previous winning scores is displayed. The user is asked their name, and the scoreboard is searched to see if they have won before. The user inputs whether or not there are two players. If there are, then the second player also puts his name in and his name is searched. Then, the turns begin.

With each turn, the user is prompted to enter ‘r’ to roll the dice. Once the user does so, the five dice are displayed. The user is asked if he would like to reroll. If he enters y, he may choose which dice to reroll. If he enters n, then he is prompted to enter which category he would like to score in.

He chooses which dice to reroll by entering y’s and n’s, with y meaning reroll the die and n meaning keep the die the same. The dice are displayed again, with some of them rerolled, and the user is asked if he would like to reroll. The process repeats itself once more.

The final numbers are displayed, along with the available category choices. There are 13 different categories: ones, twos, threes, fours, fives, sixes, three of a kind, four of a kind, small straight, large straight, full house, chance, and Yahtzee. The user chooses a category by typing and entering it, and his dice are scored for that category. Then, the second player’s turn starts, in the same fashion (if there is a second player.)

Here is how the scoring for each category works:

Ones: score is the sum of all the ones rolled

Twos: score is the sum of all the twos rolled

Threes: score is the sum of all the threes rolled

Fours: score is the sum of all the fours rolled

Fives: score is the sum of all the fives rolled

Sixes: score is the sum of all the sixes rolled

Three of a Kind: if there are three of a kind, score is the sum of all dice. Otherwise, the score is zero.

Four of a Kind: if there are four of a kind, score is sum of all dice. Otherwise, the score is zero.

Small Straight: if four consecutive numbers are rolled, score is 30. Otherwise, the score is zero.

Large Straight: if five consecutive numbers are rolled, score is 40. Otherwise, the score is zero.

Full House: if three of one number and two of another are rolled, score is 25. Otherwise, the score is zero.

Chance: score is sum of all dice

Yahtzee: if five of the same number is rolled, score is 50. Otherwise, the score is zero.

Summary of Development

Lines of Code: 531

This took about a week to develop, not counting the week I spent developing Project 1. This project is built off of Project 1, and some of the code is the same. However, this project contains many more concepts and is a completed game. It has almost 200 lines of code more than Project 1, and adds the inclusion of functions and arrays. The textbook was very helpful in developing this project, and the hardest part was figuring out how to add vectors/arrays to the program and allowing two players to play. Also, the various issues that required debugging were time-consuming.

Additions From Project 1:

Most importantly, this project includes functions, arrays, and vectors. It reads in lines of information from a file and puts it in a vector. It then sorts the vector and displays its elements in descending order. It uses bubbleSort and later in the code also uses selectionSort. It uses a linearSearch function as well. It puts the majority of the game-play (rolling, choosing categories, scoring) into a function so that two players can play without there being an excessive amount of code involved. This project uses void, int, and bool functions that take many types of arguments including reference variables, arrays, and vectors.

Example Inputs with Outputs

```
Output: Scoreboard:

        Alli     290

        Gregg    276

        Emily    260

What is your name, Player One?

Input: Emily

Output: You’ve won before. This was your last score: 260

Output: Is there a second player? Enter y for yes and n for no

Input: y

Output: What is your name, Player Two?

Input: Stacy

Output: You haven’t won before. Good luck!

Output: Your turn, Emily

Enter ‘r’ to roll the dice. Make sure you only enter one character.

Input: r

Output: You rolled:

        Dice 1: 3

        Dice 2: 6

        Dice 3: 4

        Dice 4: 3

        Dice 5: 2

        Would you like to reroll some dice? Enter y for yes and n for no.

Input: 1

Output: Would you like to reroll dice 1?
```

Input: n

Output: Would you like to reroll dice 2?

Input: n

Output: Would you like to reroll dice 3?

Input: n

Output: Would you like to reroll dice 4?

Input: y

Output: Would you like to reroll dice 5?

Input: n

Output: Enter ‘r’ to roll the dice. Make sure you only enter one character

Input: r

Output: You rolled:

```
        Dice 1: 3
        Dice 2: 6
        Dice 3: 4
        Dice 4: 5
        Dice 5: 2
        Would you like to reroll some dice? Enter y for yes and n for no
```

Input: n

Output: What category would you like to score in?

```
These are your available categories to score in:
 ones
twos
threes
fours
fives
sixes
```

three of a kind

four of a kind

small straight

large straight

full house

chance

Yahtzee

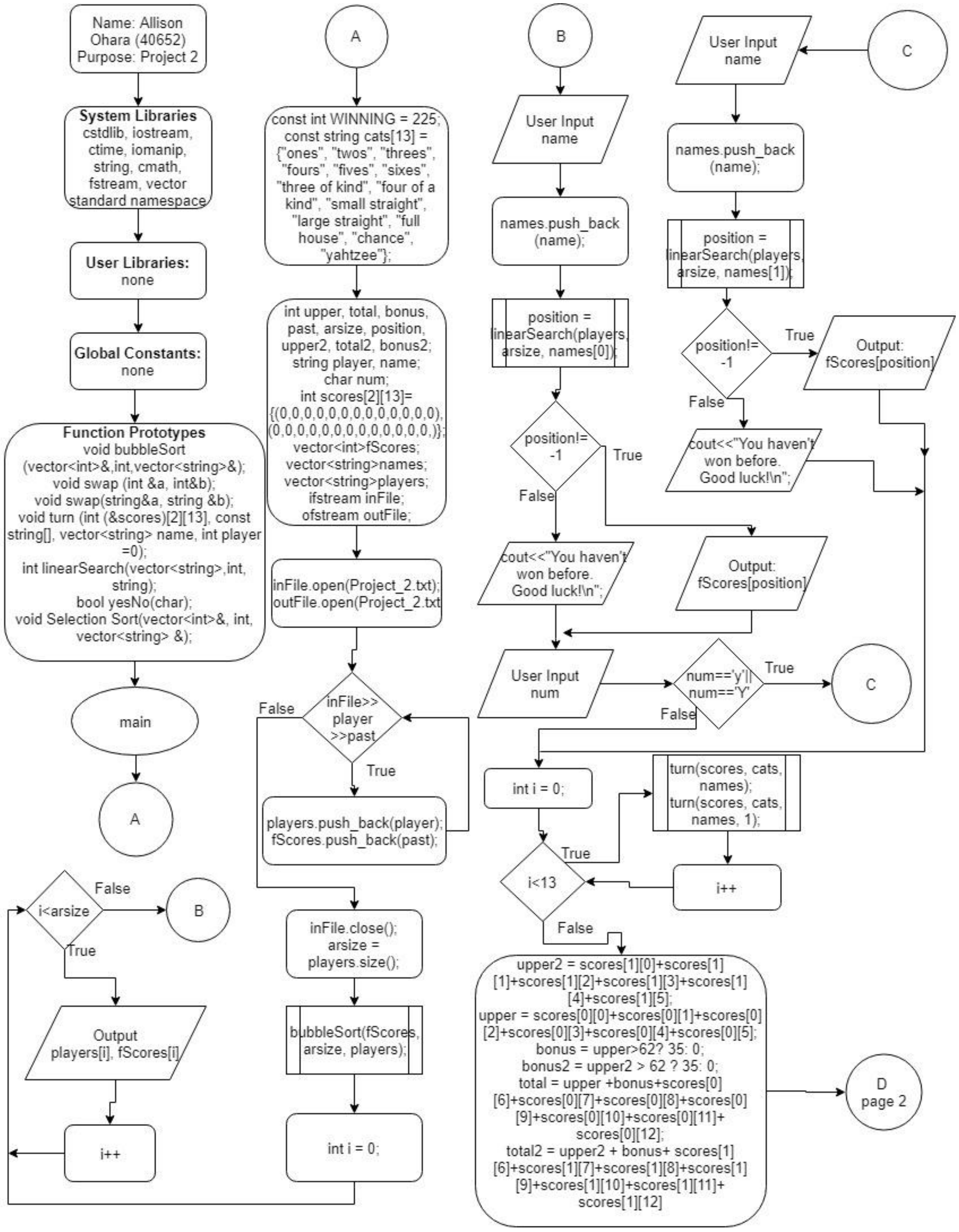
Input: large straight

Output: Category: large straight   Score: 40

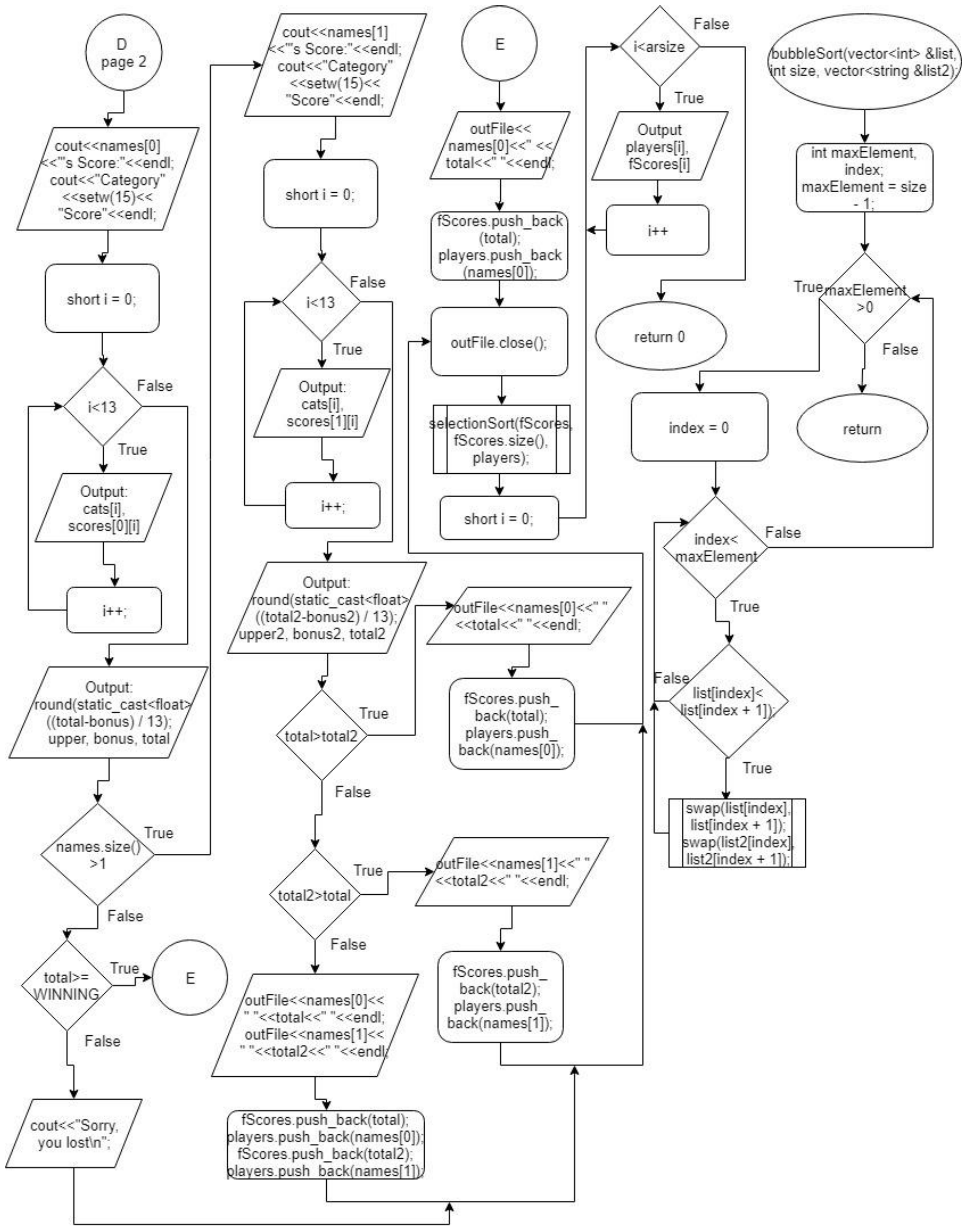
Output: Your turn, Stacy

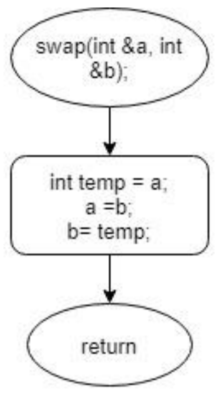
This will continue in the same way for player two's turn, and then for the next 12 turns for both players.

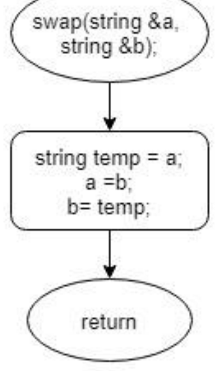
Flowcharts

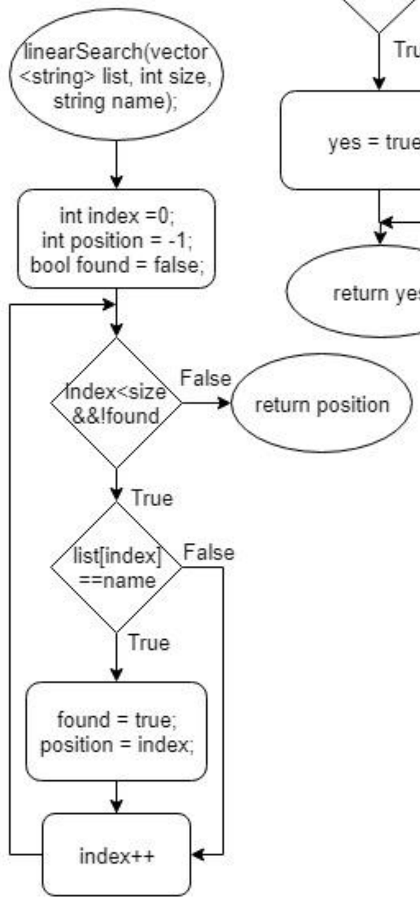


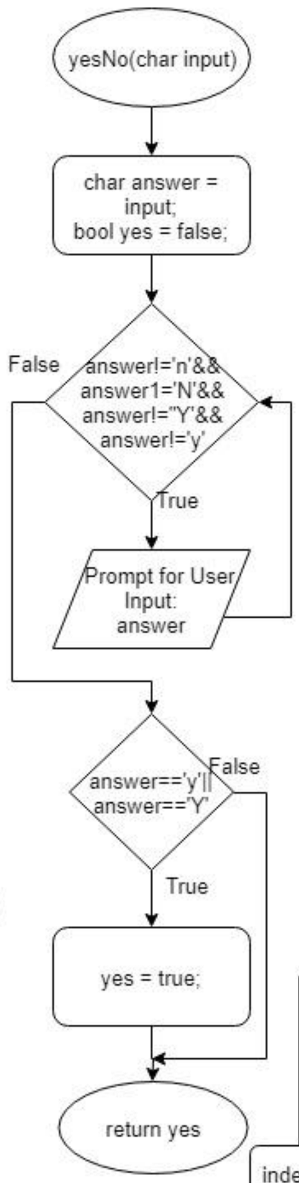
```
Name: Allison
Ohara (40652)
Purpose: Project 2

System Libraries
cstdlib, iostream,
ctime, iomanip,
string, cmath,
fstream, vector
standard namespace

User Libraries:
none

Global Constants:
none

Function Prototypes
void bubbleSort
(vector<int>&,int,vector<string>&);
void swap (int &a, int&b);
void swap(string&a, string &b);
void turn (int (&scores)[2][13], const
string[], vector<string> name, int player
=0);
int linearSearch(vector<string>,int,
string);
bool yesNo(char);
void Selection Sort(vector<int>&, int,
vector<string> &);

main

i<arsize
Output
players[i], fScores[i]
i++

const int WINNING = 225;
const string cats[13] =
{"ones", "twos", "threes",
"fours", "fives", "sixes",
"three of kind", "four of a
kind", "small straight",
"large straight", "full
house", "chance",
"yahtzee"};

int upper, total, bonus,
past, arsize, position,
upper2, total2, bonus2;
string player, name;
char num;
int scores[2][13]=
{(0,0,0,0,0,0,0,0,0,0,0,0,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,)};
vector<int>fScores;
vector<string>names;
vector<string>players;
ifstream inFile;
ofstream outFile;

inFile.open(Project_2.txt);
outFile.open(Project_2.txt

inFile>>
player
>>past

players.push_back(player);
fScores.push_back(past);

inFile.close();
arsize =
players.size();

bubbleSort(fScores,
arsize, players);

int i = 0;

User Input
name

names.push_back
(name);

position =
linearSearch(players,
arsize, names[0]);

position!=
-1

cout<<"You haven't
won before.
Good luck!\n";

Output:
fScores[position]

User Input
num

num=='y'||
num=='Y'

int i = 0;

i<13

turn(scores, cats,
names);
turn(scores, cats,
names, 1);

i++

upper2 = scores[1][0]+scores[1]
[1]+scores[1][2]+scores[1][3]+scores[1]
[4]+scores[1][5];
upper = scores[0][0]+scores[0][1]+scores[0]
[2]+scores[0][3]+scores[0][4]+scores[0][5];
bonus = upper>62? 35: 0;
bonus2 = upper2 > 62 ? 35: 0;
total = upper +bonus+scores[0]
[6]+scores[0][7]+scores[0][8]+scores[0]
[9]+scores[0][10]+scores[0][11]+
scores[0][12];
total2 = upper2 + bonus+ scores[1]
[6]+scores[1][7]+scores[1][8]+scores[1]
[9]+scores[1][10]+scores[1][11]+
scores[1][12]

User Input
name

names.push_back
(name);

position =
linearSearch(players,
arsize, names[1]);

position!=
-1

Output:
fScores[position]

cout<<"You haven't
won before.
Good luck!\n";
```

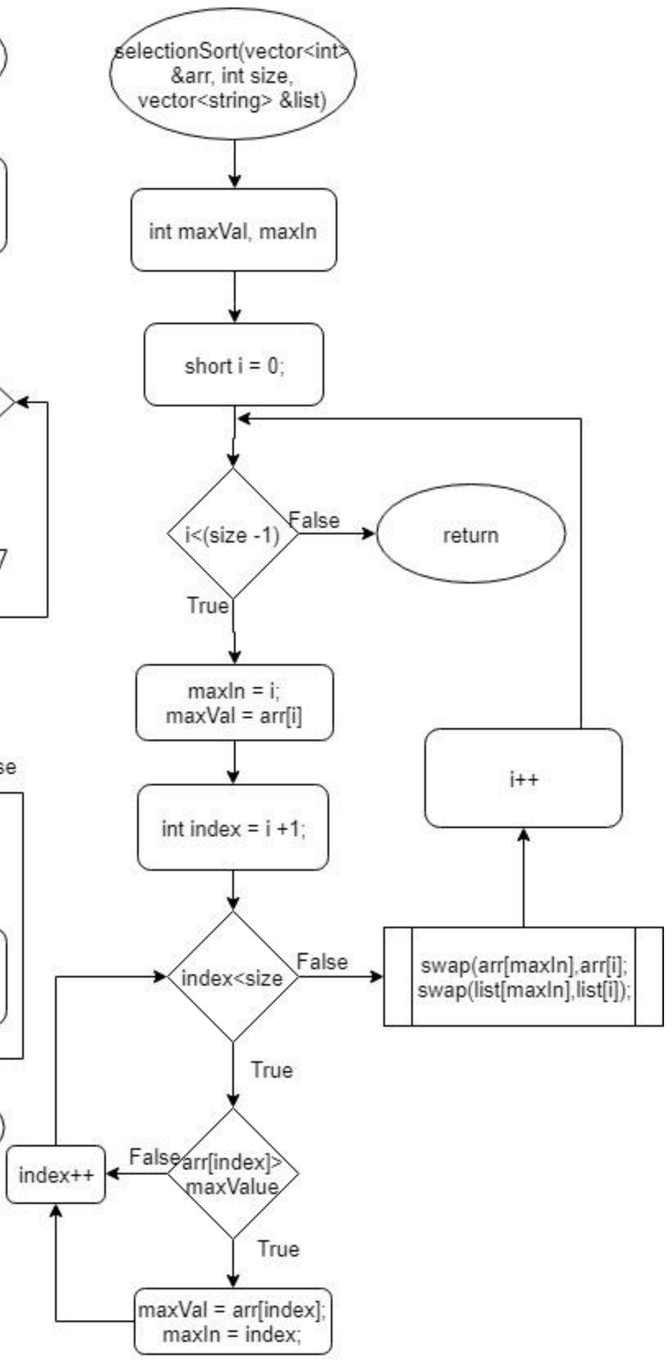

D page 2
cout<<names[0] <<"'s Score:"<<endl; cout<<"Category" <<setw(15)<< "Score"<<endl;
short i = 0;
i<13
False
True
Output: cats[i], scores[0][i]
i++;
Output: round(static_cast<float> ((total-bonus) / 13); upper, bonus, total
names.size() >1
True
False
total>= WINNING
True
E
False
cout<<"Sorry, you lost\n";
cout<<names[1] <<"'s Score:"<<endl; cout<<"Category" <<setw(15)<< "Score"<<endl;
short i = 0;
i<13
False
True
Output: cats[i], scores[1][i]
i++;
Output: round(static_cast<float> ((total2-bonus2) / 13); upper2, bonus2, total2
total>total2
True
False
total2>total
True
False
outFile<<names[0]<< " "<<total<<" "<<endl; outFile<<names[1]<< " "<<total2<<" "<<endl;
fScores.push_back(total); players.push_back(names[0]); fScores.push_back(total2); players.push_back(names[1]);
outFile<<names[0]<<" " <<total<<" "<<endl;
fScores.push_ back(total); players.push_ back(names[0]);
outFile<<names[1]<<" " <<total2<<" "<<endl;
fScores.push_ back(total2); players.push_ back(names[1]);
E
outFile<< names[0]<<" " << total<<" "<<endl;
fScores.push_back (total); players.push_back (names[0]);
outFile.close();
selectionSort(fScores, fScores.size(), players);
short i = 0;
i<arsize
False
True
Output players[i], fScores[i]
i++
return 0
bubbleSort(vector<int> &list, int size, vector<string &list2);
int maxElement, index; maxElement = size - 1;
maxElement >0
True
False
return
index = 0
index< maxElement
False
True
list[index]< list[index + 1]);
False
True
swap(list[index], list[index + 1]); swap(list2[index], list2[index + 1]);

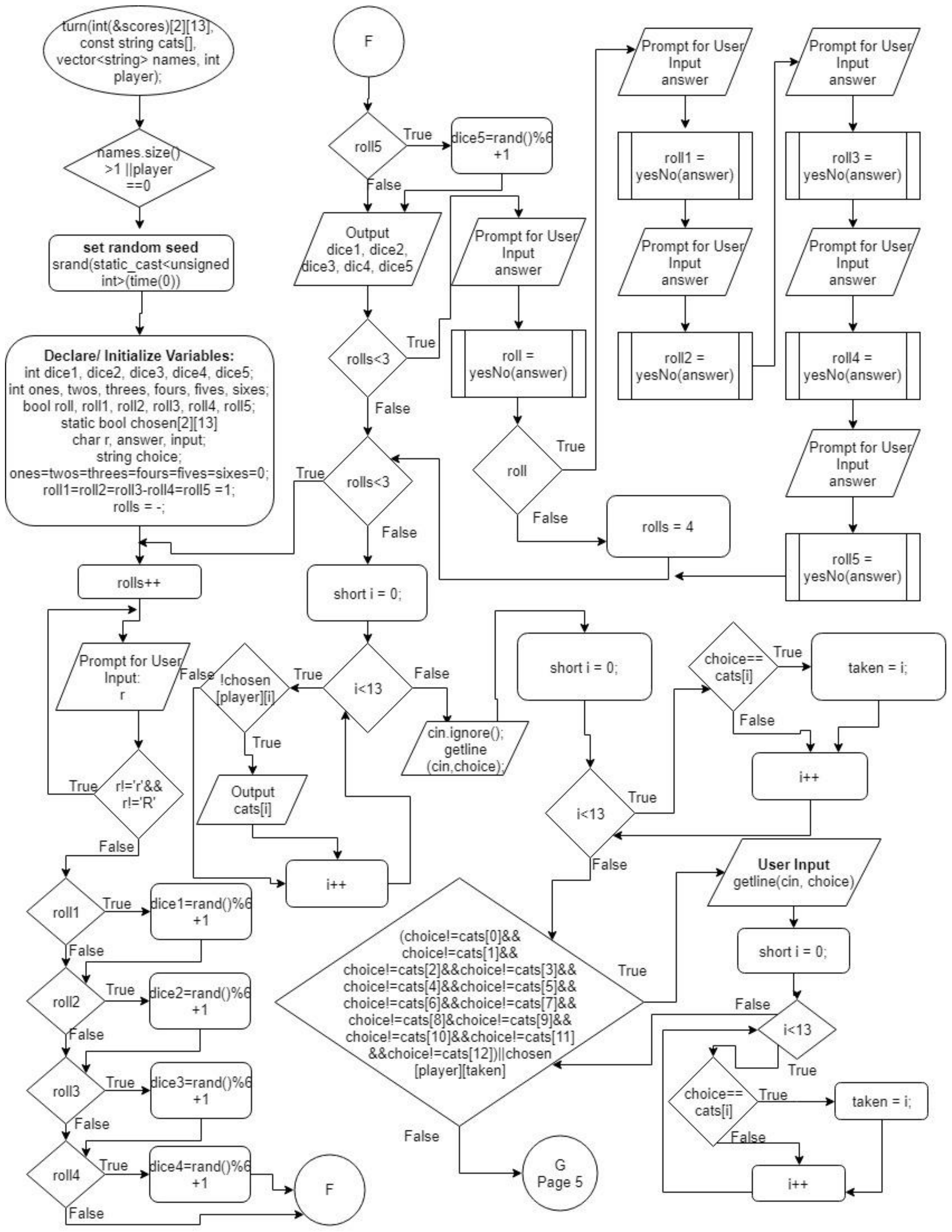

swap(int &a, int &b);
int temp = a; a =b; b= temp;
return

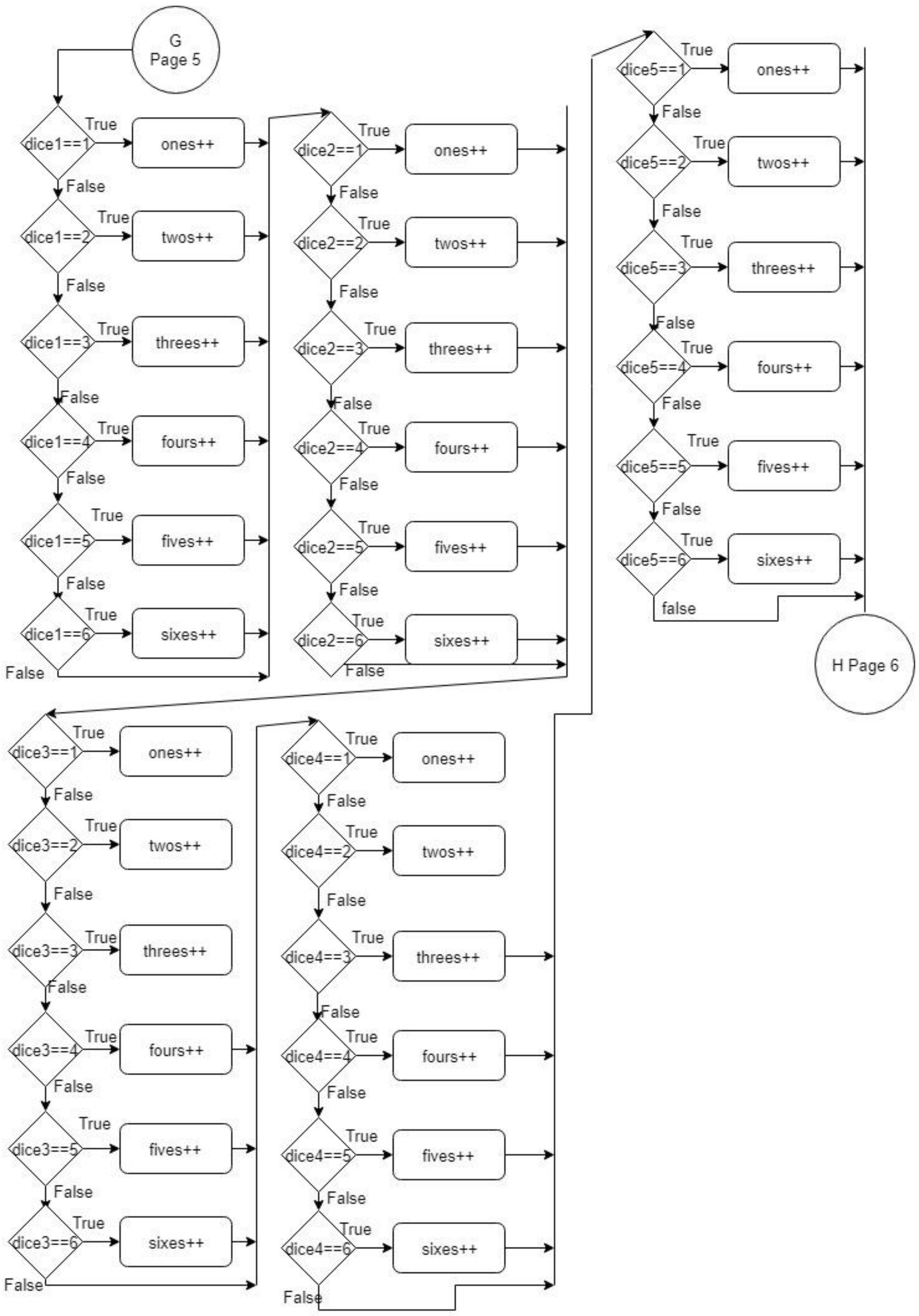

swap(string &a, string &b);
string temp = a; a =b; b= temp;
return

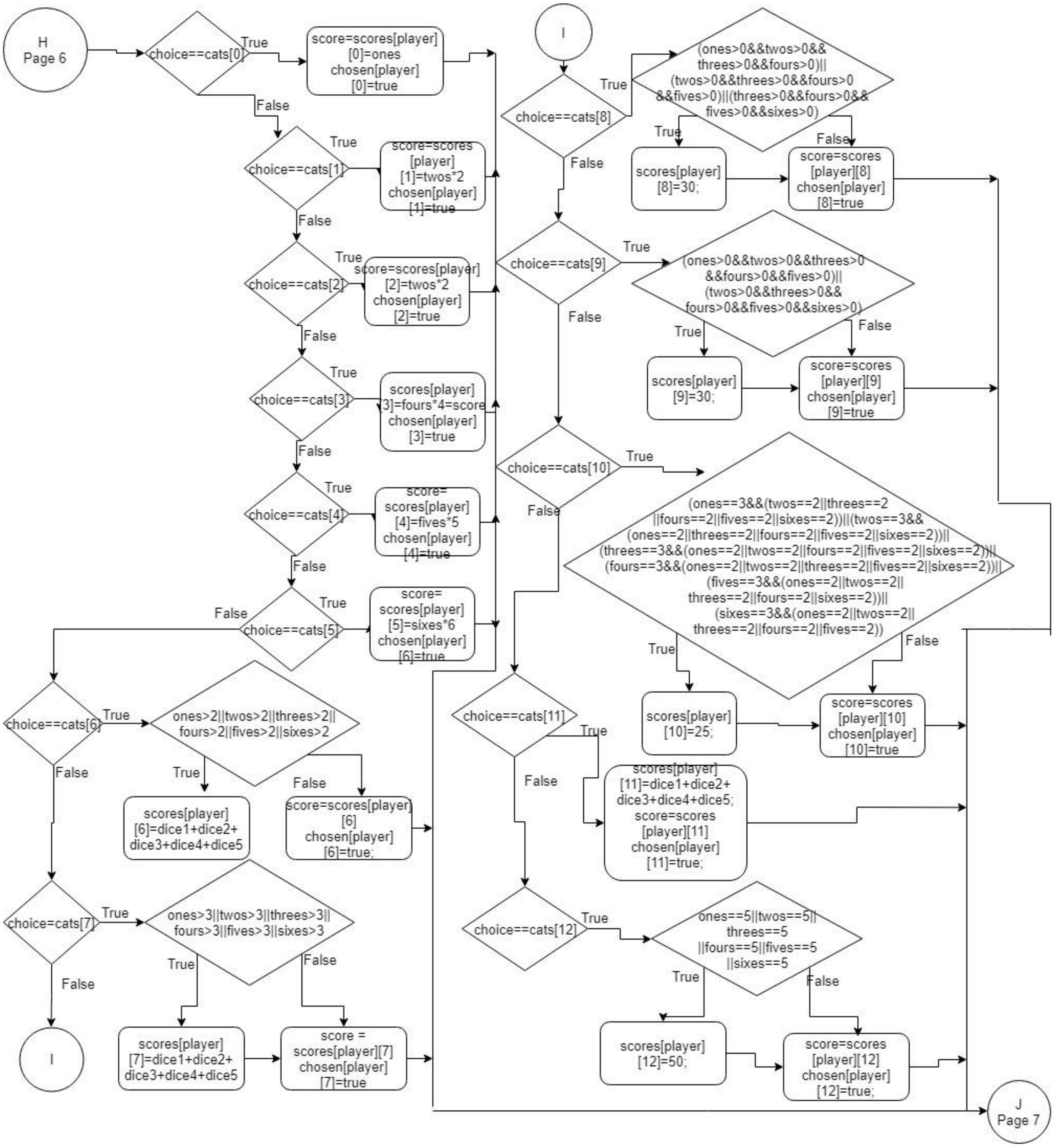

linearSearch(vector <string> list, int size, string name);
int index =0; int position = -1; bool found = false;
Index<size &&!found
False
return position
True
list[index] ==name
False
True
found = true; position = index;
index++

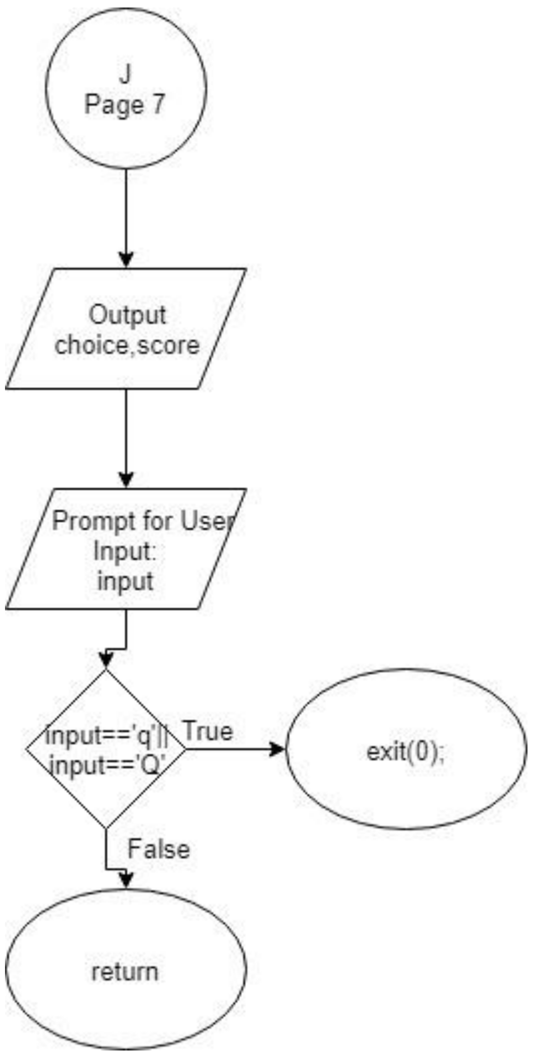

yesNo(char input)
char answer = input; bool yes = false;
False
answer!='n'&& answer1='N'&& answer!="Y'&& answer!='y'
True
Prompt for User Input: answer
answer=='y'|| answer=='Y'
False
True
yes = true;
return yes


selectionSort(vector<int> &arr, int size, vector<string> &list)
int maxVal, maxIn
short i = 0;
i<(size -1)
False
return
True
maxIn = i; maxVal = arr[i]
int index = i +1;
index<size
False
swap(arr[maxIn],arr[i]; swap(list[maxIn],list[i]);
i++
True
arr[index]> maxValue
False
index++
True
maxVal = arr[index]; maxIn = index;


turn(int(&scores)[2][13], const string cats[], vector<string> names, int player);
names.size() >1 ||player ==0
set random seed srand(static_cast<unsigned int>(time(0))
Declare/ Initialize Variables: int dice1, dice2, dice3, dice4, dice5; int ones, twos, threes, fours, fives, sixes; bool roll, roll1, roll2, roll3, roll4, roll5; static bool chosen[2][13] char r, answer, input; string choice; ones=twos=threes=fours=fives=sixes=0; roll1=roll2=roll3-roll4=roll5 =1; rolls = -;
rolls++
Prompt for User Input: r
r!='r'&& r!='R'
True
False
roll1
dice1=rand()%6 +1
roll2
dice2=rand()%6 +1
roll3
dice3=rand()%6 +1
roll4
dice4=rand()%6 +1
F
F
roll5
dice5=rand()%6 +1
Output dice1, dice2, dice3, dic4, dice5
rolls<3
rolls<3
short i = 0;
i<13
!chosen [player][i]
Output cats[i]
i++
cin.ignore(); getline (cin,choice);
Prompt for User Input answer
roll = yesNo(answer)
roll
rolls = 4
Prompt for User Input answer
roll1 = yesNo(answer)
Prompt for User Input answer
roll2 = yesNo(answer)
Prompt for User Input answer
roll3 = yesNo(answer)
Prompt for User Input answer
roll4 = yesNo(answer)
Prompt for User Input answer
roll5 = yesNo(answer)
short i = 0;
i<13
choice== cats[i]
taken = i;
i++
(choice!=cats[0]&& choice!=cats[1]&& choice!=cats[2]&&choice!=cats[3]&& choice!=cats[4]&&choice!=cats[5]&& choice!=cats[6]&&choice!=cats[7]&& choice!=cats[8]&choice!=cats[9]&& choice!=cats[10]&&choice!=cats[11] &&choice!=cats[12])||chosen [player][taken]
User Input getline(cin, choice)
short i = 0;
i<13
choice== cats[i]
taken = i;
i++
G Page 5


G Page 5
dice1==1
True
ones++
False
dice1==2
twos++
dice1==3
threes++
dice1==4
fours++
dice1==5
fives++
dice1==6
sixes++
dice2==1
dice2==2
dice2==3
dice2==4
dice2==5
dice2==6
dice3==1
dice3==2
dice3==3
dice3==4
dice3==5
dice3==6
dice4==1
dice4==2
dice4==3
dice4==4
dice4==5
dice4==6
dice5==1
dice5==2
dice5==3
dice5==4
dice5==5
dice5==6
false
H Page 6


H
Page 6
choice==cats[0]
True
score=scores[player][0]=ones
chosen[player][0]=true
False
choice==cats[1]
True
score=scores[player][1]=twos*2
chosen[player][1]=true
False
choice==cats[2]
True
score=scores[player][2]=twos*2
chosen[player][2]=true
False
choice==cats[3]
True
scores[player][3]=fours*4=score
chosen[player][3]=true
False
choice==cats[4]
True
score=scores[player][4]=fives*5
chosen[player][4]=true
False
choice==cats[5]
True
score=scores[player][5]=sixes*6
chosen[player][6]=true
False
choice==cats[6]
True
ones>2||twos>2||threes>2||fours>2||fives>2||sixes>2
True
scores[player][6]=dice1+dice2+dice3+dice4+dice5
False
score=scores[player][6]
chosen[player][6]=true;
False
choice==cats[7]
True
ones>3||twos>3||threes>3||fours>3||fives>3||sixes>3
True
scores[player][7]=dice1+dice2+dice3+dice4+dice5
False
score = scores[player][7]
chosen[player][7]=true
False
I
I
choice==cats[8]
True
(ones>0&&twos>0&&threes>0&&fours>0)||(twos>0&&threes>0&&fours>0&&fives>0)||(threes>0&&fours>0&&fives>0&&sixes>0)
True
scores[player][8]=30;
False
score=scores[player][8]
chosen[player][8]=true
False
choice==cats[9]
True
(ones>0&&twos>0&&threes>0&&fours>0&&fives>0)||(twos>0&&threes>0&&fours>0&&fives>0&&sixes>0)
True
scores[player][9]=30;
False
score=scores[player][9]
chosen[player][9]=true
False
choice==cats[10]
True
(ones==3&&(twos==2||threes==2||fours==2||fives==2||sixes==2))||(twos==3&&(ones==2||threes==2||fours==2||fives==2||sixes==2))||(threes==3&&(ones==2||twos==2||fours==2||fives==2||sixes==2))||(fours==3&&(ones==2||twos==2||threes==2||fives==2||sixes==2))||(fives==3&&(ones==2||twos==2||threes==2||fours==2||sixes==2))||(sixes==3&&(ones==2||twos==2||threes==2||fours==2||fives==2))
True
scores[player][10]=25;
False
score=scores[player][10]
chosen[player][10]=true
False
choice==cats[11]
True
scores[player][11]=dice1+dice2+dice3+dice4+dice5;
score=scores[player][11]
chosen[player][11]=true;
False
choice==cats[12]
True
ones==5||twos==5||threes==5||fours==5||fives==5||sixes==5
True
scores[player][12]=50;
False
score=scores[player][12]
chosen[player][12]=true;
J
Page 7


J
Page 7
Output choice,score
Prompt for User Input: input
input=='q'|| input=='Q'
True
exit(0);
False
return

Pseudo-code

```
/*
         * File:   main.cpp
         * Author: Allison Ohara
         * Project 2 Pseudocode
         * Updated on February 7, 2018, 12:00 p.m.
         */
        //System Libraries
        // cstdlib, used for rand
        //iostream
        //ctime, used to seed srand()
        //iomanip,used to format output with setw
        //string,used for strings
        //cmath,used for round() function
        //fstream,used for file I/O
        //vector,used for vectors


        //standard namespace


        //User Libraries


        //Global Constants


        //Function Prototypes
        //bubbleSort to sort vectors
        //swap (integers)
        //swap (strings)
        //function "turn" for each round of play in game
        //linearSearch to search for a string in a vector
        //function to turn character into a boolean
        //selectionSort to sort vectors


        //int main
            //declare variables and constants
```

```
//score needed to win
//constants for the 13 yahtzee scoring categories


//upper totals, bonuses, total scores, size of vector, and position
//initialize two dimensional array with scores all at zero


//vectors to hold the past scores, past names, current names

//ifstream and ofstream


//opening message


//open the in file
//open existing out file
//take names and scores from file and place them in vectors
//close the in file


//find the size the array needs to be by finding the size of the vector
//sort the vectors in descending order using the bubble method
//display the score board


//get the name of the player
//put the name of the player in a vector
//search for the player's name in the score board
//if player has won before, display previous score


//see if there is a second player
//if so, get second player's name and search for it in Scoreboard
//search for the player's name in the score board


//use for loop to allow both players to play 13 rounds
    //use turn function for player one
    //use turn function for player two
    //find upper score totals
    //if the upper total is greater than 62, than user receives a 35 point bonus
    //calculate total scores


    //output all categories and the scores they received for player one
    //calculate and output average score for all scoring categories (using float and math
library)

    //if there are two players, display player two's score
        //output all categories and the scores they received for player two
        //see who won and save winning score in file
        //if player one's score is greater than that of player two, player one wins
            //save score/name in file to be displayed at beginning of next game
            //add name and score to vector with past scores

        //otherwise if player two's score is greater than that of player one, player two wins
            //save score/name in file to be displayed at beginning of next game
            //add name and score to to vector with past scores

    //finally, if they tie, then they both wins and both scores/names are saved in the file
        //both scores and names are also added to the vector with past scores and names


    //if there is only one player, he wins if gets a score of 225 or better
    //if he wins, add his name and score to the scorebaord

    //close the file


    //sort updated scoreboard vectors that include winning player
    //display the updated scoreboard
    //end the program


//initialize the bubbleSort function
    //declare ints maxElement and index
    //see if each element is greater than the next
    //if so, then switch the two elements
    //keep doing this so that the vector is in descending order
    //switch the names in the parallel vector so they match up with their scores

//initialize the swap functions
```

```
    //these functions switch two values (ints or strings) in a vector and are used in the sort
functions


//initialize the linearSearch function
    //declare variables: index, position, found
    //search for the argument "name" in the vector and return the position of "name" if found


//initialize the turn function
//this function represents one round of play for one player
    //if there are two players or it is player one's turn, do the turn
        //seed the random number generator
        //declare variables
        //numbers rolled on dice
        //number of each number rolled in each round
        //whether or not each category has already been chosen
        //whether or not each die should be rolled again
        //array showing whether each category has been used or not
        //use a do while loop to allow user to roll dice
            //start at 0 for each round
            //roll all the dice at beginning of each round
            //set number of rolls to 0 for beginning of each round
            //roll the dice up to three times
            //wait until user inputs r to roll dice
            //roll the dice, all of them at first and then whichever one the user wants to the
next 2 times
                //get a random number 1-6 for die 1
                //get a random number 1-6 for die 2
                //get a random number 1-6 for die 3
                //get a random number 1-6 for die 4
                //get a random number 1-6 for die 5
            //output rolled numbers


            //allow user to roll again if he hasn't rolled 3 times yet
                //user input to roll again


                    //allow user to choose which dice to reroll
                //if they don't want to reroll, end the do while loop
        //list categories not already chosen


        //allow user to input category choice


        //while loop to validate user input


        //see how many of each number was rolled on the dice


        //score the roll according to the user inputted choice, and cross the category off
available category list

            //if category choice is ones, score is how many ones were rolled
            //cross category off list of available categories

            //if category choice is twos, score is sum of all twos rolled
            //cross category off list of available categories

            //if category choice is threes, score is sum of all threes rolled
            //cross category off list of available categories


            //if category choice is fours, score is sum of all fours rolled
            //cross category off list of available categories


            //if category choice is fives, score is sum of all fives rolled
            //cross category off list of available categories


            //if category choice is sixes, score is sum of all sixes rolled
            //cross category off list of available categories


            //if category choice is three of a kind, see if there are three of a kind
            //if so, score is sum of all die. Otherwise, score is zero.
            //cross category off list of available categories

            //if category choice is four of a kind, see if there are four of a kind
            //if so score is sum of all die. Otherwise, score is zero.
            //cross category off list of available categories

            //if category choice is small straight, see if a small straight was rolled
            //if so, score is 30. Otherwise, score is zero.
            //cross category off list of available categories


            //if category choice is large straight, see if a large straight was rolled
            //If so, score is 40. Otherwise, score is zero.
            //cross category off list of available categories


            //if category choice is full house, see if a full house was rolled
            //if so, score is 25. Otherwise, score is zero
            //cross category off list of available categories


            //if category choice is chance, score is sum of all die
            //cross category off list of available categories


            //if category choice is yahtzee, see if five of the same number was rolled
            //if so, score is 50. Otherwise score is zero
            //cross category off list of available categories
        //output score
//end function




//initialize yesNo function
    //change character argument into a boolean to return
    //while the character is neither y nor n, get another character
    //if the character is y, return true, otherwise return false


//initialize selectionSort function
    //declare variables maxIn and maxVal
    //organize the vector in descending order
    //organize the parallel vector with the names likewise
```

Code

```
/*
         * File:   main.cpp
         * Author: A llison Ohara
         * Project 2: Yahtzee 9.0 (continued from project 1)
         * Main additions (from 8.0): inclusion of all topics required in checklist, more comments,
         * working one and two player modes, debugging, etc.
         * Updated on February 7, 2018, 10:43 a.m.
         */
        //System Libraries
        #include <cstdlib>//used for rand
        #include <iostream>
        #include <ctime>//used to seed srand()
        #include <iomanip>//used to format output with setw
        #include <string>//used for strings
        #include <cmath>//used for round() function
        #include <fstream>//used for file I/O
        #include <vector> //used for vectors
        using namespace std;


        //User Libraries


        //Global Constants


        //Function Prototypes
        void bubbleSort(vector<int>&, int, vector<string>&);
        void swap(int &a, int &b);
        void swap(string &a, string &b);
        void turn(int (&scores)[2][13], const string[], vector<string> name, int player = 0);
        int linearSearch(vector<string>, int, string);
        bool yesNo(char);
        void selectionSort(vector<int> &, int, vector<string> &);


        int main(int argc, char** argv) {
            //declare variables and constants
```

```
    //score needed to win
    const int WINNING = 225;
    //constants for the 13 yahtzee scoring categories
    const string cats[13] = {"ones", "twos", "threes", "fours", "fives", "sixes", "three of a
kind",
        "four of a kind", "small straight", "large straight", "full house", "chance",
"yahtzee"};
    //upper totals, bonuses, total scores, size of vector, and position
    int upper, total, bonus, past, arsize, position;
    int upper2, total2, bonus2;
    string player, name;
    char num;
    //initialize two dimensional array with scores all at zero
    int scores[2][13] = {
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
    }; //initialize at zero
    //vectors to hold the past scores, past names, current names
    vector<int>fScores;
    vector<string>names;
    vector<string>players;
    ifstream inFile;
    ofstream outFile;
    //initialize bools at false so all categories are available
    //initialize category scores at zero
    //opening message
    cout << "You are playing the game of Yahtzee!\n";
    cout << "Good luck!\n";
    inFile.open("Project_2.txt"); //open the in file
    outFile.open("Project_2.txt", ios::app); //open existing out file
    //take names and scores from file and place them in vectors
    while (inFile >> player >> past) {
        players.push_back(player);
        fScores.push_back(past);
    }
    inFile.close(); //close the file
    //find the size the array needs to be by finding the size of the vector
    arsize = players.size();
    //sort the vectors in descending order
    bubbleSort(fScores, arsize, players);
    //display the score board
    cout << "Score Board" << endl;
    for (int i = 0; i < arsize; i++)
```

```
        cout << setw(8) << players[i] << setw(10) << fScores[i] << endl;
    //get the name of the player
    cout << "What is your name, Player One?" << endl;
    cin>>name;
    //put the name of the player in a vector
    names.push_back(name);
    //search for the player's name in the score board
    position = linearSearch(players, arsize, names[0]);
    if (position != -1) {
        //if player has won before, display previous score
        cout << "You've won before. This was your last score: " << fScores[position] << endl;
    } else
        cout << "You haven't won before. Good luck!\n";
    //see if there is a second player
    cout << "Is there a second player? Enter y for yes and any other character for no\n";
    cin>>num;
    //if so, get second player's name and search for it in Scoreboard
    if (num == 'y' || num == 'Y') {
        cout << "What is your name, Player Two?" << endl;
        cin>>name;
        names.push_back(name);
        //search for the player's name in the score board
        position = linearSearch(players, arsize, names[1]);
        if (position != -1) {
            cout << "You've won before. This was your last score: " << fScores[position] <<
endl;
        } else
            cout << "You haven't won before. Good luck!\n";
    }
    //do 13 rounds
    for (int i = 0; i < 13; i++) { //for loop
        //first player's turn
        turn(scores, cats, names);
        //second player's turn
        turn(scores, cats, names, 1);
    }
    upper2 = scores[1][0] + scores[1][1] + scores[1][2] + scores[1][3] + scores[1][4] +
scores[1][5]; //find upper score total
    upper = scores[0][0] + scores[0][1] + scores[0][2] + scores[0][3] + scores[0][4] +
scores[0][5];
    bonus = upper > 62 ? 35 : 0; //if the upper total is greater than 62, than user receives a
35 point bonus
    bonus2 = upper2 > 62 ? 35 : 0;
    total = upper + scores[0][6] + scores[0][7] + scores[0][8] + scores[0][9] + scores[0][10] +
scores[0][11] + scores[0][12] + bonus; //find total score
    total2 = upper2 + scores[1][6] + scores[1][7] + scores[1][8] + scores[1][9] + scores[1][10]
+ scores[1][11] + scores[1][12] + bonus2; //find total score
    //output all categories and the scores they received
    cout << names[0] << "'s Score: " << endl;
    //output all categories and the scores they received
    cout << "Category " << setw(15) << "Score" << endl;
    for (short i = 0; i < 13; i++)
        cout << setw(15) << cats[i] << setw(10) << scores[0][i] << endl;
    cout << endl << " Your average score in each category: ";
    //use of float is below
    cout << round(static_cast<float> (total - bonus) / 13) << endl;
    cout << " Upper Total: " << setw(11) << upper << endl;
    cout << " Bonus: " << setw(17) << bonus << endl;
    cout << endl << "Your total score: " << total << endl;
    //if there are two players, display player two's score
    if (names.size() > 1) {
        cout << names[1] << "'s Score:\n";
        //output all categories and the scores they received
        cout << "Category " << setw(15) << "Score" << endl;
        for (short i = 0; i < 13; i++)
            cout << setw(15) << cats[i] << setw(10) << scores[1][i] << endl;
        cout << endl << " Your average score in each category: ";
        //use of float is below
        cout << round(static_cast<float> (total2 - bonus2) / 13) << endl;
        cout << " Upper Total: " << setw(11) << upper2 << endl;
        cout << " Bonus: " << setw(17) << bonus2 << endl;
        cout << endl << "Your total score: " << total2 << endl;
        //see who won and save winning score in file
        if (total > total2) {
            cout << "Congrats " << names[0] << "! You won!\n";
            cout << "Sorry, " << names[1] << ". You lost.\n";
            //save score in file. to be displayed at beginning of next game
            outFile << names[0] << " " << total << " " << endl;
            fScores.push_back(total);
            players.push_back(names[0]);
        } else if (total2 > total) {
            cout << "Congrats " << names[1] << "! You won!\n";
            cout << "Sorry, " << names[0] << ". You lost.\n";
            //save score in file. to be displayed at beginning of next game
            outFile << names[1] << " " << total2 << " " << endl;
            fScores.push_back(total2);
            players.push_back(names[1]);
        } else {
            cout << "Congratulations " << names[0] << " and " << names[1] << " you tied!\n";
            cout << "Both of your names will be saved to the scoreboard.\n";
            outFile << names[0] << " " << total << " " << endl;
            outFile << names[1] << " " << total2 <<" " << endl;
            fScores.push_back(total);
            players.push_back(names[0]);
            fScores.push_back(total2);
            players.push_back(names[1]);
        }
    } else {
        //if there is only one player, he wins if gets a score of 225 or better
        //if he wins, add his name and score to the scorebaord
        if (total >= WINNING) {
            cout << "Congratulations! You got a score of 225 or better and won one-player
version\n";
            outFile << names[0] << " " << total << " " << endl;
            fScores.push_back(total);
            players.push_back(names[0]);
        } else
            cout << "Sorry; you lost.\n";
    }
    outFile.close();
    //sort updated scoreboard arrays that include winning player
    selectionSort(fScores, fScores.size(), players);
    //display the updated scoreboard
    cout << "Updated Scoreboard:\n";
    for (short i = 0; i <= arsize; i++) {
        cout << setw(8) << players[i] << setw(10) << fScores[i] << endl;
    }
    //end the program
    return 0;
}


//initialize the bubbleSort function


void bubbleSort(vector<int> &list, int size, vector<string> &list2) {
    //declare ints maxElement and index
    //see if each element is greater than the next
    //if so, then switch the two elements
    //keep doing this so that the vector is in descending order
    //switch the names in the parallel vector so they match up with their scores
    int maxElement;
    int index;
    for (maxElement = size - 1; maxElement > 0; maxElement--) {
        for (index = 0; index < maxElement; index++) {
            if (list[index] < list[index + 1]) {
                swap(list[index], list[index + 1]);
                swap(list2[index], list2[index + 1]);
            }
        }
    }
}
//initialize the swap function


void swap(int &a, int &b) {
    //switch the two function arguments
    int temp = a;
    a = b;
    b = temp;
}
//initialize the other swap function, switching two string elements


void swap(string &a, string&b) {
    string temp = a;
    a = b;
    b = temp;
}
//initialize the linearSearch function


int linearSearch(vector<string> list, int size, string name) {
    //declare variables: index, position, found
    int index = 0;
    int position = -1;
    bool found = false;
    //search for the argument "name" in the vector and return the position of "name" if found
    while (index < size&& !found) {
        if (list[index] == name) {
            found = true;
            position = index;
        }
        index++;
    }
    return position;
}
//initialize the turn function
//this function represents one round of play for one player


void turn(int (&scores)[2][13], const string cats[], vector<string> names, int player) {
    //if there are two players or it is player one's turn, do the turn
    if (names.size() > 1 || player == 0) {
        cout << "Your turn " << names[player] << endl;
        //seed the random number generator
        srand(static_cast<unsigned int> (time(0))); //type casting
        //declare variables
        char input;
        int dice1, dice2, dice3, dice4, dice5; //numbers rolled on dice
        int ones, twos, threes, fours, fives, sixes, rolls, score; //number of each number
rolled in each round
        //whether or not each category has already been chosen
        bool roll, roll1, roll2, roll3, roll4, roll5; //whether or not each die should be rolled
again
        static bool chosen [2][13]; //array showing whether each category has been used or not
       int taken;
        char r, answer;
        string choice;
        ones = twos = threes = fours = fives = sixes = 0; //start at 0 for each round
        roll1 = roll2 = roll3 = roll4 = roll5 = 1; //roll all the dice at beginning of each
round
        rolls = 0; //set number of rolls to 0 for beginning of each round
        //roll the dice up to three times
        do {
            rolls++;
            do {
                cout << "Enter 'r' to roll the dice. Make sure you only enter one character.\n";
                cin>>r;
            } while (r != 'r' && r != 'R'); //wait until user inputs r to roll dice
            //roll the dice, all of them at first and then whichever one the user wants to the
next 2 times
            if (roll1)
                dice1 = rand() % 6 + 1; //get a random number 1-6
            if (roll2)
                dice2 = rand() % 6 + 1; //get a random number 1-6
            if (roll3)
                dice3 = rand() % 6 + 1; //get a random number 1-6
            if (roll4)
                dice4 = rand() % 6 + 1; //get a random number 1-6
            if (roll5)
                dice5 = rand() % 6 + 1; //get a random number 1-6
            //output rolled numbers
            cout << "You rolled:\n";
            cout << "Dice 1:" << setw(3) << dice1 << endl;
            cout << "Dice 2:" << setw(3) << dice2 << endl;
            cout << "Dice 3:" << setw(3) << dice3 << endl;
            cout << "Dice 4:" << setw(3) << dice4 << endl;
            cout << "Dice 5:" << setw(3) << dice5 << endl;
            //allow user to roll again if he hasn't rolled 3 times yet
            if (rolls < 3) {
                cout << "Would you like to reroll some dice? Enter y for yes and n for no\n";
                cin>>answer; //user input to roll again
                roll = yesNo(answer);
                if (roll) {
                    //allow user to choose which dice to reroll
                    cout << "Would you like to reroll die 1? Enter y for yes and n for no\n";
                    cin>>answer;
                    roll1 = yesNo(answer);
                    cout << "Would you like to reroll die 2? Enter y for yes and n for no\n";
                    cin>>answer;
                    roll2 = yesNo(answer);
                    cout << "Would you like to reroll die 3? Enter y for yes and n for no\n";
                    cin>>answer;
                    roll3 = yesNo(answer);
                    cout << "Would you like to reroll die 4? Enter y for yes and n for no\n";
                    cin>>answer;
                    roll4 = yesNo(answer);
                    cout << "Would you like to reroll die 5? Enter y for yes and n for no\n";
                    cin>>answer;
                    roll5 = yesNo(answer);
                } else { //if they don't want to reroll, end the do while loop
                    rolls = 4;
                }
            }
        } while (rolls < 3);
        cout << "What category would you like to score in?\n";
        cout << "These are your available categories to score in:\n";
```

```
        //list categories not already chosen
        for (short i = 0; i < 13; i++) {
            if (!chosen[player][i])
                cout << cats[i] << endl;
        }
        //allow user to input category choice
        cin.ignore();
        getline(cin, choice);
        for(short i = 0; i<13; i++){
            if(choice == cats[i]){
                taken = i;
            }
        }
        //while loop to validate user input
        while ((choice != cats[0] && choice != cats[1] && choice != cats[2] && choice != cats[3]
&& choice != cats[4] && choice != cats[5] && choice != cats[6] &&
                choice != cats[7] && choice != cats[8] & choice != cats[9] && choice != cats[10]
&& choice != cats[11] && choice != cats[12])||chosen[player][taken]) {
            cout << "Invalid category choice. Enter again.\n";
            getline(cin, choice);
            for(short i = 0; i<13; i++){
                if(choice == cats[i]){
                    taken = i;
                }
            }
        }
        //for loop to make sure chosen category hasn't already been chosen

        //see how many of each number was rolled on the dice
        switch (dice1) {
            case 1:ones++;
                break;
            case 2: twos++;
                break;
            case 3:threes++;
                break;
            case 4: fours++;
                break;
            case 5: fives++;
                break;
            case 6: sixes++;
                break;
        }
```

```
switch (dice2) {
    case 1:ones++;
        break;
    case 2: twos++;
        break;
    case 3:threes++;
        break;
    case 4: fours++;
        break;
    case 5: fives++;
        break;
    case 6: sixes++;
        break;
}
switch (dice3) {
    case 1:ones++;
        break;
    case 2: twos++;
        break;
    case 3:threes++;
        break;
    case 4: fours++;
        break;
    case 5: fives++;
        break;
    case 6: sixes++;
        break;
}
switch (dice4) {
    case 1:ones++;
        break;
    case 2: twos++;
        break;
    case 3:threes++;
        break;
    case 4: fours++;
        break;
    case 5: fives++;
        break;
    case 6: sixes++;
        break;
}
switch (dice5) {
```

```
                case 1:ones++;
                    break;
                case 2:twos++;
                    break;
                case 3:threes++;
                    break;
                case 4: fours++;
                    break;
                case 5: fives++;
                    break;
                case 6: sixes++;
                    break;
            }
            //score the roll according to the user inputted choice, and cross the category off
available category list
            //dependent if / if else


            if (choice == cats[0]) {
                scores[player][0] = ones; //score is how many ones were rolled
                score = scores[player][0];
                chosen[player][0] = true; //category is chosed, don't list as available (this occurs
at the end of each category)
            } else if (choice == cats[1]) {
                scores[player][1] = twos * 2; //score is sum of all twos rolled
                score = scores[player][1];
                chosen[player][1] = true;
            } else if (choice == cats[2]) {
                scores[player][2] = threes * 3; //score is sum of all three rolled
                score = scores[player][2];
                chosen[player][2] = true;
            } else if (choice == cats[3]) {
                scores[player][3] = fours * 4; //score is sum of all fours rolled
                score = scores[player][3];
                chosen[player][3] = true;
            } else if (choice == cats[4]) {
                scores[player][4] = fives * 5; //score is sum of all fives rolled
                score = scores[player][4];
                chosen[player][4] = true;
            } else if (choice == cats[5]) {
                scores[player][5] = sixes * 6; //score is sum of all sixes
                score = scores[player][5];
                chosen[player][5] = true;
```

```
        } else if (choice == cats[6]) {
            if (ones > 2 || twos > 2 || threes > 2 || fours > 2 || fives > 2 || sixes > 2) //see
if there are three or more of one number
                scores[player][6] = dice1 + dice2 + dice3 + dice4 + dice5; //if there are three
of a kind, score is sum of all die
            //otherwise score is zero
            score = scores[player][6];
            chosen[player][6] = true;
        } else if (choice == cats[7]) {
            if (ones > 3 || twos > 3 || threes > 3 || fours > 3 || fives > 3 || sixes > 3)//see
if there are four or more of one number
                scores[player][7] = dice1 + dice2 + dice3 + dice4 + dice5; //if there are four
of a kind, score is sum of all die
            //otherwise, score is zero
            score = scores[player][7];
            chosen[player][7] = true;
        } else if (choice == cats[8]) {
            if ((ones > 0 && twos > 0 && threes > 0 && fours > 0) ||
                    (twos > 0 && threes > 0 && fours > 0 && fives > 0) ||
                    (threes > 0 && fours > 0 && fives > 0 && sixes > 0))//see if there is a 1-2-
3-4 straight
                scores[player][8] = 30; //if so, score is 30
            //otherwise score is zero
            score = scores[player][8];
            chosen[player][8] = true;
        } else if (choice == cats[9]) {
            if ((ones > 0 && twos > 0 && threes > 0 && fours > 0 && fives > 0) || (twos > 0 &&
threes > 0 && fours > 0 && fives > 0 && sixes > 0))//see if there is a 1-2-3-4-5 straight
                scores[player][9] = 40; //if so score is 40
            //otherwise score is zero
            score = scores[player][9];
            chosen[player][9] = true;
        } else if (choice == cats[10]) {
            //see if there is 3 of one number and two of another
            //if so, score is 25
            //otherwise, score is zero
            if ((ones == 3 && (twos == 2 || threes == 2 || fours == 2 || fives == 2 || sixes ==
2)) ||
                    (twos == 3 && (ones == 2 || threes == 2 || fours == 2 || fives == 2 || sixes
== 2)) ||
                    (threes == 3 && (ones == 2 || twos == 2 || fours == 2 || fives == 2 || sixes
== 2)) ||
                    (fours == 3 && (ones == 2 || twos == 2 || threes == 2 || fives == 2 || sixes
== 2)) ||
                    (fives == 3 && (ones == 2 || twos == 2 || threes == 2 || fours == 2 || sixes
== 2)) ||
                    (sixes == 3 && (ones == 2 || twos == 2 || threes == 2 || fours == 2 || fives
== 2)))
                scores[player][10] = 25;
            score = scores[player][10];
            chosen[player][10] = true;
        } else if (choice == cats[11]) {
            scores[player][11] = dice1 + dice2 + dice3 + dice4 + dice5; //score is sum of all
die
            score = scores[player][11];
            chosen[player][11] = true;
        } else if (choice == cats[12]) {
            if (ones == 5 || twos == 5 || threes == 5 || fours == 5 || fives == 5 || sixes == 5)
                scores[player][12] = 50; //score is 50 if there are five of the same number
            //otherwise score is zero
            score = scores[player][12];
            chosen[player][12] = true;
        }
        //output score
        cout << "Category: " << choice << "       Score: " << score << endl;
        cout << "If you would like to quit the game, enter q. If you would like to continue
playing, enter any other character.\n";
        cin>>input;
        if (input == 'q' || input == 'Q')
            exit(0);
    }
}
//initialize yesNo function


bool yesNo(char input) {
    //change character argument into a boolean to return
    //while the character is neither y nor n, get another character
    //if the character is y, return true, otherwise return false
    char answer = input;
    bool yes = false;
    while (answer != 'n' && answer != 'N' && answer != 'y' && answer != 'Y') {
        cout << "Invalid entry. Enter y for yes or n for no.";
        cin>>answer;
    }
    if (answer == 'y' || answer == 'Y')
        yes = true;
    return yes;
}
//initialize selectionSort function


void selectionSort(vector<int>&arr, int size, vector<string>&list) {
    //declare variables maxIn and maxVal
    int maxVal, maxIn;
    //organize the vector in descending order
    //organize the parallel vector with the names likewise
    for (short i = 0; i < (size - 1); i++) {
        maxIn = i;
        maxVal = arr[i];
        for (int index = i + 1; index < size; index++) {
            if (arr[index] > maxVal) {
                maxVal = arr[index];
                maxIn = index;
            }
        }
        swap(arr[maxIn], arr[i]);
        swap(list[maxIn], list[i]);
    }
}
```

Completed Check-off Sheets

Cross Reference from Project 1

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
2	2	cout			
	3	libraries	10-17	5	iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals	41-44		No variables in global area, failed project!
	5	Identifiers	41-44		
	6	Integers	41, 42, 46	1	
	7	Characters	44, 253	1	
	8	Strings	43, 260	1	
	9	Floats No Doubles	140, 127	1	Using doubles will fail the project, floats OK!
	10	Bools	257, 509	1	
	11	Sizeof *****			
	12	Variables 7 characters or less	254,255		All variables <= 7 characters
	13	Scope ***** No Global Variables			
	14	Arithmetic operators	452,441		
	15	Comments 20%+	247, 250, 256	2	Model as pseudo code seen in pseudo code file
	16	Named Constants	36, 38		All Local, only Conversions/Physics/Math in Global area
	17	Programming Style ***** Emulate			Emulate style in book/in class repositiory
3	1	cin	81, 325		
	2	Math Expression	429, 117		
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting	127, 251	1	
	6	Multiple assignment *****			
	7	Formatting output	124, 129	1	
	8	Strings	260, 43	1	
	9	Math Library	127, 140	1	All libraries included have to be used
	10	Hand tracing ******			
4	1	Relational Operators	445, 451		
	2	if	445, 248	1	Independent if
	4	If-else	172/177	1	
	5	Nesting	333/334/335	1	
	6	If-else-if	420/424/428	1	
	7	Flags *****			
	8	Logical operators	420, 445	1	
	11	Validating user input	327, 333	1	
	13	Conditional Operator	115,116	1	
	14	Switch	347, 361	1	
5	1	Increment/Decrement	266, 202	1	
	2	While	66, 327	1	
	5	Do-while	265-315	1	
	6	For loop	333, 525	1	
	11	Files input/output both	66, 174	2	
	12	No breaks in loops ******			Failed Project if included
****** Not required to show			Total	30	

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	25-31	4	Always use prototypes
	5	Pass by Value	28, 30, 504	4	
	8	return	516, 241	4	A value from a function
	9	returning boolean	516	4	
	10	Global Variables		XXX	Do not use global variables -100 pts
	11	static variables	258	4	
	12	defaulted arguments	28	4	
	13	pass by reference	25, 26, 221, 213	4	
	14	overloading	26/27, 213/221	5	
	15	exit() function	499	4	
7		Arrays			
	1 to 6	Single Dimensioned Arrays	38	3	
	7	Parallel Arrays	46/258	2	Parallel vectors: 51/53
	8	Single Dimensioned as Function Arguments	28/28/109	2	
	9	2 Dimensioned Arrays	46, 258	2	Emulate style in book/in class repositiory
	12	STL Vectors	51-53	2	
		Passing Arrays to and from Functions	109,111, 246	5	
		Passing Vectors to and from Functions	246, 520, 182	5	
8		Searching and Sorting Arrays			
	3	Bubble Sort	194-210, 74	4	
	3	Selection Sort	182, 520-537	4	
	1	Linear or Binary Search	228-242, 85	4	
****** Not required to show			Total	70	Other 30 points from Proj 1 first sheet tab