Project 10

due at midnight on Wed Dec 16 (60 points)

For this project, we will implement a simplified form of the dice game called Yahtzee. It works a bit like Poker – you roll five dice, and then you can discard and re-roll some of them. You try to build 'hands' like five of a kind, full house, two pair, etc.

Below are transcripts of a few games using my solution, and below that is a skeleton of a solution. You just need to fill in the function definitions. Read the documentation in that given code carefully, and have fun playing your game!

Call your program `p10dice.cpp` and submit to this dropbox for project 10.

Game one

```
WELCOME TO Yahtzee!
Dice:
  (a) 2
  (b) 4
  (c) 1
  (d) 3
  (e) 5
Nothing!
Which to roll again? abc
Dice:
  (a) 2
  (b) 4
  (c) 6
  (d) 3
  (e) 5
Nothing!
Which to roll again? abcde
Dice:
  (a) 2
  (b) 2
  (c) 4
  (d) 1
  (e) 1
Two pair.
GAME OVER
```

Game two

```
WELCOME TO Yahtzee!
Dice:
  (a) 4
  (b) 4
  (c) 2
  (d) 3
  (e) 6
One pair.
Which to roll again? cde
Dice:
  (a) 4
  (b) 4
  (c) 4
  (d) 5
  (e) 4
Four of a kind.
Which to roll again? d
Dice:
  (a) 4
  (b) 4
  (c) 4
  (d) 3
  (e) 4
Four of a kind.
GAME OVER
```

Game three

```
WELCOME TO Yahtzee!
Dice:
  (a) 1
  (b) 1
  (c) 6
  (d) 2
  (e) 3
One pair.
Which to roll again? cde
Dice:
  (a) 1
  (b) 1
  (c) 2
  (d) 3
  (e) 6
```

```
One pair.
Which to roll again? cde
Dice:
  (a) 1
  (b) 1
  (c) 4
  (d) 2
  (e) 3
One pair.
GAME OVER
```

Game four

```
WELCOME TO Yahtzee!
Dice:
  (a) 3
  (b) 5
  (c) 1
  (d) 1
  (e) 2
One pair.
Which to roll again? abe
Dice:
  (a) 1
  (b) 4
  (c) 1
  (d) 1
  (e) 4
Full house.
Which to roll again?
Dice:
  (a) 1
  (b) 4
  (c) 1
  (d) 1
  (e) 4
Full house.
GAME OVER
```

p10given.cpp

```
// Yahtzee game -- YOUR NAME HERE
#include <iostream>
#include <vector>
```

```
#include <ctime>
#include <cstdlib>
using namespace std;

// Function prototypes: see documentation for each below.
int  roll_one_die();
vector<int> roll_all_dice(int num);
void roll_these_again(vector<int>& dice, string which);
void print_dice(vector<int> dice);
void print_best_hand(vector<int> dice);
bool n_of_a_kind(vector<int> tally, int n);
int  num_pairs(vector<int> tally);

/* Main program: you shouldn't change this very much.
 * You may temporarily replace what's here with some
 * test code.
 */
int main()
{
    cout << "WELCOME TO Yahtzee!" << endl;
    srand(time(NULL));          // Initialize PRNG
    const int NUM_DICE = 5;
    vector<int> dice = roll_all_dice(NUM_DICE);
    int rolls_left = 2;
    while(true)
    {
        print_dice(dice);
        print_best_hand(dice);
        if(rolls_left == 0)
        {
            break;
        }
        cout << "Which to roll again? ";
        string selected;
        getline(cin, selected);
        roll_these_again(dice, selected);
        rolls_left--;
    }
    cout << "GAME OVER" << endl;
    return 0;
}

/* This function will simulate rolling one 6-sided
 * die, returning a single random number between
 * 1 and 6.
 */
int roll_one_die()
{
    return 0; // TODO
}

/* This function takes takes `num`, the number of dice,
 * and generates a vector containing that many random
 * dice rolls.
 */
vector<int> roll_all_dice(int num)
{
    vector<int> dice;
    // TODO
    return dice;
}

/* This function should print the values of all the dice
 * in the given vector, with a lower-case letter (a-e)
 * beside each one so we can refer to it. For example:
 *   (a) 6
 *   (b) 3
 *   (c) 5
 *   (d) 2
 *   (e) 5
 */
void print_dice(vector<int> dice)
{
    // TODO
}

/* This function will roll selected dice again. The string
 * `which` is what the user typed, containing a sequence
 * of lower-case letters in the range a-e. The die in the
 * vector corresponding to each of those should be re-rolled.
 * WARNING: be careful to error-check, so that you don't end
 * up trying to re-roll a die that is out of bounds!
 */
void roll_these_again(vector<int>& dice, string which)
{
    // TODO
}

/* This function should compute a TALLY of the values in
 * the dice vector. Then it can use that tally along with
 * the two helper functions below to determine the best
 * hand. The ordering of hands from best to worst is:
 *   - 5 of a kind (aka Yahtzee)
 *   - Full house (3 of one kind, and 2 of another)
 *   - Four of a kind
 *   - Three of a kind
 *   - Two pair
 *   - One pair
 */
void print_best_hand(vector<int> dice)
{
    // TODO
}

/* This function returns true/false, as to whether the
 * given `tally` represents a set of dice with exactly
 * `n` of a kind. It can be reused to detect 5 of a kind,
 * 4 of a kind, etc.
 */
bool n_of_a_kind(vector<int> tally, int n)
{
    // TODO
    return false;
}

/* This function counts the number of times that `2`
 * appears in the `tally` vector, which means the
 * number of pairs in the hand.
 */
int num_pairs(vector<int> tally)
{
    // TODO
    return 0;
}
```