CS 7B - Fall 2017 - Assignment 4: The Game of Set. Due 12/4/17

The Game of Set Rules (this is a hyperlink to https://www.setgame.com/set/puzzle_rules) should be studied carefully if you are still unsure how the game works. Also, practice the online version at https://www.nytimes.com/crosswords.

Given: a set of objects called "`Setcards`", each of which has four attributes: `number, color, shade, shape`, and each attribute can occur in three different ways.

A `Set` is a group of three cards where each attribute is either all the same or all different.

The goal of this project is create a program to have a human user play against the computer.

I have written an implementation in which each of a `Setcard`'s attributes is represented as a `string`:
{`"One", "Two", "Three"`} for the `number`
{`"green", "red", "purple"`} for the `color`
{`"blank", "striped", "filled"`} for the `shade`
{`"oval", "diamond", "wiggle"`} for the `shape`
These comprise the `Setcard` class's private attributes.

Create a `cards.h` header file with the prototypes of the objects and functions you'll use. Here's how mine starts:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <string>
using namespace std;

//Initilize constant variables
const int SIZE = 81;

class Setcard {
public:
    Setcard(); //default constructor
    Setcard(string n, string c, string shd, string shp);
    void print(bool = true);
    string getNumber() { return number; }
    string getColor() { return color; }
    string getShade() { return shade; }
    string getShape() { return shape; }
    bool operator==(Setcard& rhs);
private:
    string number;
    string color;
    string shade;
    string shape;
};
```

In the same `cards.h` header, I created a dependent class:

```
class deckOfCards {
public:
    deckOfCards();  // Default constructor: assigns the 81 cards to deck
    void shuffle(); //shuffles the deck once all the cards are assigned
    //deal one card from the top of the deck, reshuffling the discards if needed
    Setcard dealCard();
    vector<Setcard> dealNine(); //to create the original 3x3 array
private:
Setcard* deck = new Setcard[SIZE]; // an array of cards of size SIZE
    int currentCard; //keep track of what card you are dealing with
};
```

This is not necessarily the way to do. It is not a necessarily neccessary and/or sufficient collection of objects and functions your design will require, but it is an example of typical functions and objects that would be involved in such a design.

Also include in `cards.h` the prototypes of the helper functions you use, such as some of those that I wrote:

```
// helper function prototypes
void showBoard(vector<Setcard> b);
Setcard third(Setcard s1, Setcard s2); //for any 2 cards, the 3rd that makes a Set
ostream& operator<<(ostream& os, Setcard s);

template<typename T>
void printElement(T t, const int& width);
bool isSet(const vector<Setcard>& vs); //takes a 3 tuple of Setcard, true if Set
bool gotSet(const vector<Setcard>& vs);
```

You need not structure your program to use these prototypes, but they're the some of the ones I came up with and my program works durn good...sorta.

Here's the `main()` function I used:

```
int main() {
    //Initialize scores:
    int playerScore{0}, AIScore{0};
    //create a deck of Setcards in unshuffled order
    deckOfCards deck;
    //declare Setcard called(from class) current card
    Setcard currentCard;
    //shuffle the deck that you just initalized
    deck.shuffle();
    //determine how many cards you want to print out to the user
    //right now it is two because we decided that each player will get two cards when they start
    vector<Setcard> board = deck.dealNine();
    //showBoard(board);

    while(1) { //game loop
        showBoard(board);
        if(playerTurn(board,deck)) ++playerScore;
            //cout << "\nScore Player: " << playerScore << ", Computer: " << AIScore;
        if(computerTurn(board,deck)) ++AIScore;
        cout << "\nScore Player: " << playerScore << ", Computer: " << AIScore;
```

```
    }
}
```

This `main()` function is in a file called `playSet.cpp` together with the definitions of
`bool playerTurn(vector<Setcard>& board, deckOfCards& deck)`
and
`bool computerTurn(vector<Setcard>& board, deckOfCards& deck)`.

Build the functionality of the program gradually. Start with very simple basic features and adding more, as needed...test each baby step on the way.

Here's a suggestion for the baby steps:
1. Write the default and parameterized constructors for the `Setcard` card.

2. Write a `print()` function for the `Setcard` class that will print a card as, say, "`Two green filled diamonds`".

3. Modify `Setcard's print()` function to take a `bool abbrev` that is `true` by default and, if `true` prints the abbreviated version `2gfd` for the above mentioned card.

4. Adopt the ideas from above to overload the insertion operator for the `Setcard` class with `ostream& operator<<(ostrea`
`os, Setcard s)`.

5. Write the default constructor for the `deckOfCards` class. It should set the four attributes to one of three string values to create a full set of 81 `Setcards` in the `deckOfCards`.
Check that everything so far works by printing out the entire deck of `Setcards` in order using both the long and abbreviated versions of the `Setcards`.

6. Write `void deckOfCards::shuffle()` to give a good shuffle and then check that the deck is shuffled by printing it all out.

7. Write the `Setcard deckOfCards::dealCard()` function and check it by dealing out more than 81 cards.

8. Get

```
template<typename T>
void printElement(T t, const int& width) {
    cout << right << setw(width) << setfill(' ') << t;
}
```

to work with

```
void showBoard(const vector<Setcard>& b) {
    std::cout.width(6);
    cout<< endl << endl << setw(9) << " A  " << setw(8) << " B  " << setw(8) << " C  " << "\n";
    for(int i = 0; i < 9; ++i) {
        if(i%3==0) cout << char(49+(i+1)/3); //row numbers
        printElement(b[i],5);// << '\t' << setw(6) << dealCard() << dealCard()<< dealCard() << endl
        if((i+1)%3==0) cout << "\n";
    }
}
```

to get some well formatted display of the `Setcard`s on the board.
9. Write `bool isSet(const vector<Setcard>& vsc)` which return returns `true` if a given vector of three `Setcard`s is a Set.
10. Write `bool gotSet(const vector<Setcard>& vsc)` which return returns `true` if a given vector of nine

`Setcard`s has a Set.
Remember to keep testing your code as you build.
11. Write `Setcard third(Setcard s1, Setcard s2)` which will return the unique `Setcard` that will complete a Set for a given pair of `Setcard`s. This is for the computer to search for a Set in a given board.
12. Write `bool playerTurn(vector<Setcard>& board, deckOfCards& deck)` which return `true` if a player either correctly determines that a board has no Set or identifies a Set in a board. If the player incorrectly determines that there is no Set or identifies a triplet that is not a Set, the function returns `false`.
13. Write `bool ComputerTurn(vector<Setcard>& board, deckOfCards& deck)` which searches for a Set in a given board and returns `true` and reports what it found if it finds one or returns `false`.

Here's some output from an early prototype that stranded the user if there's no Set:

```
      A       B       C
1    2gfd    3rbo    1rfo
2    2gso    3pfd    3gbo
3    1psd    3rfw    1gfo

Enter the column letters and row numbers of a Set: C 1 A 2 C 2
0
Computer found: 2gso, 3gbo, 1gfo
Score Player: 0, Computer: 1

      A       B       C
1    2gfd    3rbo    1rfo
2    2psd    3pfd    1rsd
3    1psd    3rfw    3gbd

Enter the column letters and row numbers of a Set: A 1 B 2 C 3
0
Score Player: 0, Computer: 1

      A       B       C
1    2gfd    3rbo    1rfo
2    2psd    3pfd    1rsd
3    1psd    3rfw    3gbd

Enter the column letters and row numbers of a Set:
```

The program correctly determines the user has entered code for a triplet that doesn't comprise a Set. Then the computer does find a Set. Score one for the computer and none for us. When the computer found a Set, new cards are dealt in the position of cards where the Set was called. Then we get a board with no Set! So be sure to provide a way of recognizing there is no Set in a board and redealing the board in that case.

Here's an interaction showing the more advanced game play:

```
      A       B       C
1    3gbo    1rbo    2pfw
2    2rfw    3pso    2psd
3    1rfo    3rsd    3gsw

Did you find a set? y or n: y

Enter the column letters and row numbers of a Set: B 1 A 2 B 3
```

```
1
You found 1rbo  2rfw  3rsd
No Sets are present.  Redealing:
Score Player: 1, Computer: 0

      A       B       C
1    1psd    2pbo    1pbw
2    2rbo    1rbd    1gbd
3    3rfw    3gbw    1pso

Did you find a set? y or n: y

Enter the column letters and row numbers of a Set: B 1 B 2 B 3
1
You found 2pbo  1rbd  3gbw
Computer found: 1psd, 2gbo, 3rfw
Score Player: 2, Computer: 1

      A       B       C
1    3gfw    2pbd    1pbw
2    2rbo    3rfo    1gbd
3    1rfw    1gfd    1pso

Did you find a set? y or n: y

Enter the column letters and row numbers of a Set: C 1 B 3 C 3
0
That wasn't a Set.
Computer found: 1gbd, 1rfw, 1pso
Score Player: 2, Computer: 2

      A       B       C
1    3gfw    2pbd    1pbw
2    2rbo    3rfo    1gfo
3    3pbd    1gfd    3pbo

Did you find a set? y or n:
```