ITSE 1430- Fall 2016
Program Set #2

See C# Grading /Program Guide Sheet for directions and grading/submission information.

Note: Create a console application program for each of the following problems. Do not create a windows application (form).

1. One of the more interesting sports found at the Olympic games is Table Tennis (a.k.a. ping pong). A game is won by being the first player to win 11 points, and be at least 2 points ahead of his or her opponent. If both players have won 10 points, then the first player to get a 2-point lead wins the game. A match is the best of 7 games (4 wins) in both men and women's singles events. Write a C# program using methods to simulate an Olympic ping pong singles match.

Assume that the two players in a ping pong game are each assigned a skill level from 0 to 10. The higher the skill level, the more likely that player is to win a particular point. For example, if player 1 and player 2 have identical skill levels, say 6.5 and 6.5, each is equally likely to win a given point. However, if player 1 has a skill level of 6.5 and player 2 has a skill level of 3.25, then player 1 is twice as likely to win a given point. For example, in Python:

```python
def pingPongPoint(level1, level2):
    """Simulates a single point in a ping pong game and
       returns the winning player number (1 or 2).
       level1 and level2 are real numbers between 0 and 10
       indicating a play level for each player (10 is best)."""
    pick = (level1+level2)*random.random()
    if pick < level1:
        return 1
    else:
        return 2
```

The method above (`pingPongPoint`) simulates a single point in a ping pong game and returns the winning player number (1 or 2). The method should have two inputs real numbers between 0 and 10 indicating a play level for each player (10 is the best). Other method to use in the program:

- `pingPongGame()`-Simulates a complete game of ping pong by calling `pingPongPoint` repeatedly. The method should have three inputs, the number of points required to win a game, the skill level of player 1, and the skill level of player 2 (in that order). It should print the score after each point. Remember a player has to win a game by at least two points. You will need to think carefully how to address this in your method.

Let the user input the players by number (Player 1, etc.) as well as their skill levels. The program should output players, each point, games won, and the overall match winner. Output should be user friendly.

Name the program: `PingPongXX.cs`, where XX are your initials.

2. The transpose of a matrix is obtained by interchanging the rows with the corresponding columns. Or more formally: The transpose A^T of a matrix A can be obtained by reflecting the elements along its main diagonal. Repeating the process on the transposed matrix returns the elements to their original position. For example, the matrix below:

```
12 13 14     Can be transposed to:     12 15 18
15 16 17                               13 16 19
18 19 11                               14 17 11
```

Write a C# program that lets the user input the size and the values of the matrix from the keyboard and outputs the corresponding transposed matrix. Check for sizes up to 5 by 5, and one user-defined method must be used in the program. Output should look similar to below.

Sample Runs:

```
Enter matrix size: 3 2      Enter matrix size:  3 3
Enter the matrix:           Enter the matrix:
1 2                         12 13 14
3 4                         15 16 17
5 6                         18 19 11

The transposed matrix is:   The transposed matrix is:

1 3 5                       12 15 18
2 4 6                       13 16 19
                            14 17 11
```

Name the program: `MatrixTransposeXX.cs`, where XX are your initials.

3. One of the quirks of the English language is the lack of consistency between phonetics and spelling. Classifying names by their phonetic structure is the goal of the Soundex system. The Soundex algorithm groups together words that appear to sound alike based on their spelling. The rules of the algorithm are as follows:

1. The first letter of the encoding will always be the first letter of the string
2. Ignore the letters `A, E, H, I, O, U, W,` and `Y` except in rule #1
3. Use the table below to assign the remaining letters their encoding (discard any non-letter characters: dashes, spaces, and so on)
   ```
   1- B F P V
   2- C G J K Q S X Z
   3- D T
   4- L
   5- M N
   6- R
   ```

4. Any letters with the same encoding that appear subsequently to each other should be ignored. For example, the sequence "BB" would produce the encoding "1". The sequence "BAB" would also produce the encoding "1" since A is an ignored character.
5. The total length of the encoding must always be four. If you have more than four characters in your encoding, trim the extra characters. If you have less than four characters pad the encoding with zeroes to bring the length to four.

Write a C# program that allows the user to enter a name (convert to all uppercase) and prints its Soundex code as determined by the above algorithm. It should operate in a loop that allows the user to get codes as many names as desired before exiting. Output should look similar to below.

Sample Runs:

```
Enter surname: Chegwidden
Soundex code for CHEGWIDDEN is C235

Run again (Y/N): y

Enter surname: Zach
Soundex code for ZACH is Z200

Run again (Y/N): n
```

Name the program: `SoundexXX.cs`, where `XX` are your initials.

4 (**). Write a C# console program that plays the casino game blackjack, or 21. Blackjack is a competition between the dealer and player(s) to see who can get closest to 21 points without going over (busting).

Implementation Notes

- The player is dealt two cards face up, and the dealer (computer) initially only show one of his cards face up.
- The player has the choice to take a card (hit) or to stay with what he has.
- A hands present total is calculated by simply adding the face value of the cards.
- In blackjack the cards are valued as follows:
  - An Ace can count as either 1 or 11 (the Ace value only becomes 1 if the player would otherwise have bust).
  - The cards from 2 through 10 are valued as indicated.
  - The J, Q, and K are all valued at 10.

  The suits of the cards do not have any meaning in the game.
- The player may keep taking cards until he chooses to stay, or until his cards total more than 21. If this happens the player is deemed to have bust and losses the hand. The hand is then over.
- If the player has stayed on 21 or less it is the dealer's turn to take cards.
- The dealer must follow very fixed rules and keep taking cards until he has at least 17, and he must stop when he gets 17 or more.

- If through taking cards the dealer's total exceeds 21 he too busts. If the dealer busts and the player still remains, then the player wins the hand.
- If the dealer does not bust the winner of the hand is the person with the highest total, whether that be the dealer or the player.
- If the totals are identical a push results and the hand is deemed a draw (tie).
- The best hand in the game is called blackjack- this consists of an Ace and any card valued at 10, in any order. If the dealer gets blackjack he automatically wins the hand unless the player also has a blackjack in which case a push (draw) results.
- A standard deck of cards is used- no jokers.
- Use 3 card decks- and reshuffle decks when less than 10 cards are available.
- Use card character symbols:
    - ♣ - clubs ♠ - spades ♥ - hearts ♦ - diamonds

Modifications to above:

- Have the dealer (computer) initially only show one of his cards face up. Example:

```
Dealer's cards: [?] XX Q♠
```

Here the value of the hand is unknown. Once the player has finished his turn he will reveal his other card. Then ask user to play again.

```
Dealer's cards: [?] XX Q♠
Your cards: [11] 6♠ 5♥

 (S)-Stand or (H)-Hit or (Q)-Quit? H

You drew: [17] 6♥
(S)-Stand or (H)-Hit or (Q)-Quit? S

Dealer's cards: [20] Q♣ Q♠

Dealer wins

Game over. Good-bye.
```

- The prompt for intra-round commands must show which commands are currently available for the player. Note not all commands are available at all times.

```
(H)-Hit (S)-Stand (D)-Double Down (P)-Split (I)-Insurance (Q)-Quit
```

- Create a menu that allows the user to:
    - Input values from keyboard (like above)- demo mode
    - Input values from a randomly generated values (default)- game mode

```
Do you wish to play in (E)-Demo or (G)-Game mode? e
```

- Use a string to enter the player's name via the keyboard.

```
Enter player name: James
```

- Implement simple betting- Rules:
  - Only the player can bet
  - Bets must be made before any cards are dealt.
  - The minimum bet must be $5 and the maximum bet of $100.
  - Initial value of $100 - if the player loses, subtract the bet from the money, else if wins, add amount equal to the user's money.
  - The game should quit when the player runs out of money.
  - The player can only bet in $5 increments.
  - If the player gets a blackjack he gets two and one half his bet (3-2 odds) as opposed to the normal return for winning.

```
James you have $100 dollars.
How much do you want to bet? 100

Dealer's cards: [?] XX 7♣

Your cards: [16] A♥ 5♠
(S)-Stand or (H)-Hit or (Q)-Quit? H

Your cards: [17] A♥ 5♠ A♣
(S)-Stand or (H)-Hit or (Q)-Quit? h

Your cards: [15] A♥ 5♠ A♣ 8♠
(S)-Stand or (H)-Hit or (Q)-Quit? S

Dealer's cards: [10] 3♣ 7♣
Dealer drew: [12] 2♦
Dealer drew: [15] 3♠
Dealer drew: [19] 4♥

Dealer wins

James looses $100

Game over. Good-bye.
```

Other features to add:

- **Insurance**- When the dealer's face up card is an ace, a side bet of up to half of the original bet that the dealer's face down card is a ten (point) card, thus a blackjack for the house. If the dealer's card is a ten card, it is turned up, and the player who made the insurance bet wins and paid double the amount of their half bet- 2 to 1 payoff. If the player has blackjack as well- it is a stand-off (tie).
- **Doubling Down.** Another option open to the player is doubling his bet when the original two cards dealt total 9, 10, or 11. When the player's turn comes, he places a bet equal to the original bet, and the dealer gives him just one card, which is placed face down and is not turned up until the bets are settled at the end of the hand. With two fives, the player may split a pair, double down, or just play the hand in the regular way.
- **Splitting Pairs.** If a player's first two cards are of the same denomination, such as two jacks or two sixes, he may choose to treat them as two separate hands when his turn comes around. The amount of his original bet then goes on one of the cards, and an equal amount must be placed as a bet on

the other card. The player first plays the hand to his left by standing or hitting one or more times; only then is the hand to the right played. The two hands are thus treated separately, and the dealer settles with each on its own merits. Doubling down is allowed after a split. With a pair of aces, the player is given one card for each ace and may not draw again. Also, if a ten-card is dealt to one of these aces, the payoff is equal to the bet (not one and one-half to one, as with a blackjack at any other time)- this is not a blackjack but a regular 21.

- The dealer may never double-down, split, or take insurance.

Use user-defined methods in your program. Output should look similar to above.

Name the program: `BlackjackXX.cs`, where `XX` are your initials.