COSC 1437 (DL) - Fall 2016
Program Set #3

See 1437 Grading /Program Guide Sheet for directions and grading/submission information.

1. Bingo is a game of chance. Each player takes a Bingo card. Each card has 25 spaces organized into 5 columns and 5 rows under the "BINGO" header, as follows:

- Each space has a value between 1 and 75 selected at random, except no two spaces on a card can have the same value. The allowed range of values in each column is as follows:

  B: 1-15, I: 16-30, N: 31-45, G: 46-60, and O: 61-75

Then during game play, values between 1 and 75 are selected at random, traditionally by placing numbered balls in a basket and drawing them out one at a time. Once a number is selected, it must not be selected again in the same game. If a number on a card is selected, the player covers the corresponding space on the card. In addition, the center space (the third space down in the N column) is called FREE and is always covered. A player wins the game by covering at least one of the following patterns:

- All the spaces in any row.
- All the spaces in any column.
- All the spaces on either diagonal.
- The four corner spaces.

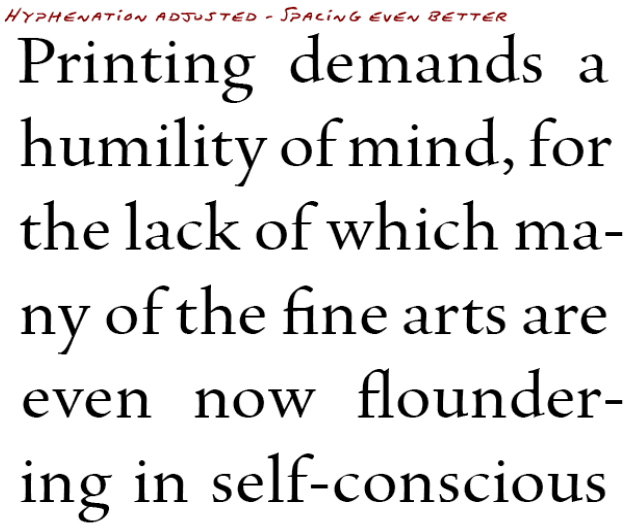
Write a C++ program that plays a simulation of an electronic Bingo game. Two players will play the game.

- Have two program options. The first option is the demo mode- a demo mode is required so the program can be graded. In demo mode, the user enters the bingo numbers (i.e. B2, O63, etc.). Separate each value with a space when guessing (inputting to screen). This will allow you to test your logic, results, etc. The second option is game mode. In this mode, the program will randomly generate the bingo numbers.
- Generate and store two legal bingo cards. The middle square, the free square, is marked with an X. The two cards are generated at start-up.
- Output the Bingo numbers called for each card.
- For both cards check if the bingo number is on the card, and if so- mark the cell with an X, and check for a winning pattern listed above. Only one winning pattern per card.
- Continue playing until at least 1 player wins. Print a message congratulating the winner, the winning card, and winning pattern (values).

At least one user-defined function must be used. Output should be user friendly.

Name the program: BingoXX.cpp, where XX are your initials.

2. Word processing programs such as MS Word spilt words across lines using hyphenation. The word processor knows how to break words into basic syllables used for hyphenating words. Write a C++ program that accepts a list of words (a consecutive string of non-whitespace characters) as input and prints each word, one per line, with hyphens inserted at each possible hyphenation point defined by the following simple rules:



- If you see the pattern `vowel-consonant-consonant-vowel`, hyphenate between the two consonants. The vowels are: `a, e, i, o, u` and `y`. The letter `y` will always be treated as a vowel in this problem.
- If you see the pattern `vowel-consonant-vowel`, hyphenate before the consonant unless the second vowel is an `e` and occurs at the end of the word.
- The following character sequences are never divided by hyphens: `"qu", "tr", "br", "str", "st", "sl", "bl", "cr", "ph", "ch"`.
- For the purpose of applying bullets 1 and 2, these are all considered to be a single consonant.
- Upper and lower-case distinctions are ignored for applying the above rules, although the case in the input word must be preserved in the output.

Must use C++ strings/functions in your program. Output should look similar to below.

Sample Run:

```
Enter the text: The rules given in this problem are a bit crude. But they represent a
good starting point.

The
ru-les
gi-ven
in
this
pro-blem
are
a
bit
cru-de.
But
they
rep-re-sent
a
good
star-ting
point.
```

Name the program: `HyphenationXX.cpp`, where `XX` are your initials.

3. During the Rio 2016 Olympics the media would frequently show a "medal count" that lists how many gold, silver, and bronze medals have been won by the athletes of each participating country. Usually, the countries would be listed in descending order by how many medals, in total, its athletes have won. Ties are broken first by considering the number of gold medals won; if those numbers are equal, the number of silver medals won is used to break the tie. If two countries match exactly in numbers of gold, silver, and bronze medals won, they are listed in alphabetical order, using their three-letter (NOC) code.
Write a C++ program that, given as input for each Olympic event the countries represented by its gold, silver, and bronze medal winners, displays a medal count as illustrated below.

Input will come from a text file which contains a positive integer n on the first line indicating how many Olympic events have been completed. Each of the next n lines contains the three-letter (NOC) codes identifying the countries represented by the gold, silver, and bronze medal winners, respectively, of one event. The output will consist of a title and a table giving the medal counts of each country. The first line of the table should have column headings G, S, B, and T (referring to the three medal colors and "Total"). Each subsequent line of the table should include a country's three-letter (NOC) code followed by the numbers of gold, silver, and bronze medals won by athletes from that country, followed by the total number of medals. Countries are to be listed in descending order by total medals won, with ties broken by number of gold medals, then by number of silver medals, and, finally, by country code (alphabetical order). Each number should be right-justified in a field of width three. It is possible for a medal count to reach two or three digits. Let the user input the file name from the keyboard. Output should look similar to below.

Sample Input File:

```
15
USA GBR DEN
DEN FRA NED
NOR NOR USA
RUS USA USA
NED RUS DEN
NOR POL NOR
CAN RUS USA
NED RUS POL
SWE CHN RUS
RUS SWE NED
USA LAT GBR
GRE POL DEN
NOR CAN CAN
NED ITA USA
JPN AUT AUS
```

Sample Run:

```
Enter the file name: medals.txt

Medal Count

      G     S     B     T
USA   2     1     4     7
RUS   2     3     1     6
NOR   3     1     1     5
NED   3     0     2     5
DEN   1     0     3     4
CAN   1     1     1     3
POL   0     2     1     3
SWE   1     1     0     2
GBR   0     1     1     2
GRE   1     0     0     1
JPN   1     0     0     1
AUT   0     1     0     1
CHN   0     1     0     1
FRA   0     1     0     1
ITA   0     1     0     1
LAT   0     1     0     1
AUS   0     0     1     1
```

Name the program: `MedalCountXX.cpp`, where XX are your initials.

4 (**). Write a C++ program that determines a winner of a 2 player poker match. A poker hand of five playing cards drawn from a standard 52-card deck has a value, based on the table below. Each card has a rank (Ace, 2–9, 10, Jack, Queen, or King) and a suit (♣, ♦, ♥, or ♠). A card is represented as a 2-character string with a rank of (A, 2–9, T, J, Q, or K) and a suit of (C, D, H, or S). Thus, "AS" represents the Ace of Spades, while "TC" represents the Ten of Clubs.

Power	Poker Hand Value	Description	Examples
10	ROYAL FLUSH	The 10, Jack, Queen, King and Ace of the same suit. This hand is the strongest of all poker hands.	A♦ K♦ Q♦ J♦ T♦
9	STRAIGHT FLUSH	Five cards in sequence. Aces can play low or high. All cards belong to the same suit. Note that Ace may not "wrap around" and play both high and low.	Q♣ J♣ T♣ 9♣ 8♣ **or** 5♦ 4♦ 3♦ 2♦ A♦
8	FOUR OF A KIND	Contains all four cards of one rank and any other (unmatched) card	9♣ 9♠ 9♦ 9♥ J♥
7	FULL HOUSE	Contains three matching cards of one rank and two matching cards of another rank	7♠ 7♥ 7♦ 4♠ 4♣
6	FLUSH	All five cards are of the same suit, but not in sequence	Q♦ 9♦ 7♦ 4♦ 3♦
5	STRAIGHT	Five cards in sequence using at least two different suits. Aces can play low or high. Note that Ace may not "wrap around" and play both high and low.	A♣ K♣ Q♦ J♠ T♠ **or** 5♠ 4♦ 3♦ 2♠ A♥
4	THREE OF A KIND	Contains three matching cards of the same rank and two unmatched cards of other ranks	Q♠ Q♥ Q♦ 7♠ 4♣
3	TWO PAIR	Contains two cards of the same rank, plus two cards ofanother rank (that match each other but not the first pair), plus any card not of either rank	T♠ T♣ 8♥ 8♣ A♦
2	ONE PAIR	Contains two cards of one rank, plus three cards which are neither of that rank nor the same as each other.	4♥ 4♠ K♠ T♦ 5♠
1	HIGH CARD	Five cards not meeting any of the above requirements.	K♥ J♥ 8♣ 7♦ 4♠

Play with two players and use a string to enter names via the keyboard. Hands will be input from the keyboard with five card representations, converted to uppercase. Each card is a string of two characters—a value followed by a suit as described earlier. If the user enters an illegal card or tries to enter the same card twice, the program will ignore the card, issue an error, and then request another card.

Check for high card winners and check for ties. Hands are ranked first by category, then by individual card ranks. That is, even the minimum qualifying hand in a certain category defeats all hands in all lower categories. The smallest two pair hand, for example, defeats all hands with just one pair or no

pair. Only between two hands in the same category are card ranks used to break ties. The highest single card in each flush or straight is used to break ties (the ace-through-five straight is the lowest straight, the ace being a low card in this context). Within two pair hands, the higher pairs are first compared. If they tie, then secondary pairs are compared, and then finally the kicker.

if equal strength	**winner determined by tie breaker**
royal flush	always a tie
straight flush	highest value card
four of a kind	highest value four of a kind
full house	highest value three of a kind
flush	highest value card; else next highest value card; else next highest, etc.
straight	highest value card
three of a kind	highest value three of a kind; else highest value lone card; else next highest value lone card
two pairs	highest value pair; otherwise next highest value pair; else highest value lone card
one pair	highest value pair; else highest value lone card; else next highest value lone card; else next highest value lone card
high card	highest value card; else next highest value card; else next highest, etc.

Your output will consist of a line of text by itself, consisting of the player's name, strongest poker hand value (in CAPITAL LETTERS) and the fully described hand like below:

```
AD KD TD QD JD:  ROYAL FLUSH, Diamonds
5D 3D 2D AD 4D:  STRAIGHT FLUSH, Diamonds, 5 high
9C 9S JH 9D 9H:  FOUR OF A KIND, 9's
7S 4C 7C 7D 4S:  FULL HOUSE, 7's full of 4's
             (NOTE: <three of kind> full of <pair>)
QD 9D 7D 4D 3D:  FLUSH, Diamonds, Q high
AC JS KC QD TS:  STRAIGHT, A high
QS 4C QH 7S QD:  THREE OF A KIND, Q's
TS 8H TC 8C AD:  TWO PAIR, T's and 8's
             (NOTE: Higher ranking pair is listed first)
4J TD 4S KS 5S:  ONE PAIR, 4's
KC 3D 2H AS JH:  HIGH CARD, K High
```

Also, change the output to reflect the card symbols:

♣ - clubs
♦ - diamonds
♥ - hearts
♠ - spades

Finally, output the winner and ask the user if he/she would like to play again (upper and lowercase).

Use functions and the C++ string class in your program. Also, only use one dimensional arrays- do not use 2D arrays, structs, or classes in this problem. A sort of any kind cannot be used in the problem as well. Output should look similar to below.

Sample Run:

```
Enter player 1 name:  James
Enter player 2 name:  Morgan

James:
Enter a card: TC
Enter a card: 2D
Enter a card: 2D
Duplicate Card. Try again.
Enter a card: 5H
Enter a card: 5C
Enter a card: 5D

Morgan:
Enter a card: KS
Enter a card: JS
Enter a card: S9
Bad Card. Try again
Enter a card: 9S
Enter a card: 2S
Enter a card: 3S

James: T♣ 2♦ 5♥ 5♣ 5♦ : THREE OF A KIND, 5's
Morgan: K♠ J♠ 9♠ 2♠ 3♠ : FLUSH, Spades, K high

Morgan wins!!

Play again (Y/N)? Y

James:
Enter a card: AD
Enter a card: JH
Enter a card: 9S
Enter a card: 5D
Enter a card: 3C

Morgan:
Enter a card: AS
Enter a card: TD
Enter a card: yC
Bad Card. Try again
Enter a card: 9C
Enter a card: 6H
Enter a card: 4H

James: A♦ J♥ 9♠ 5♦ 3♣ : HIGH CARD, J High
Morgan:  A♠ T♦ 9♣ 6♥ 4♥ : HIGH CARD, T High

James wins!!
```

```
Play again (Y/N)? y

James:
Enter a card: AD
Enter a card: JH
Enter a card: 9S
Enter a card: 6D
Enter a card: 4C

Morgan:
Enter a card: AS
Enter a card: JD
Enter a card: 9C
Enter a card: 6H
Enter a card: 4H

James:  A♦ J♥ 9♠ 6♦ 4♣ : HIGH CARD, J High
Morgan: A♠ J♦ 9♣ 6♥ 4♥ : HIGH CARD, J High

Tie- split pot!!

Play again (Y/N)? n
```

Name the program: `Poker.cpp`, where XX are your initials.