CS 202 - Fall 2017
Dr. Williams - sections 1003, 1004, and 1007
Assignment 1
Electronic copy due Wednesday, September 20 by 11:59 AM
Hard copy due Wednesday, September 20 or Thursday, September 21 in your class section

This assignment will involve creating a class called `bingocard` that implements a bingo card and related functions. The functions we are concerned with are as follows:

- Filling the card with an initial set of values.
- Marking a value & displaying a message when bingo is obtained the first time.
- Determining how many marks away a card is from making a bingo.
- Displaying the card values & marks.
- Comparing two cards to see how many values they share.
- Comparing two cards to see if they are equivalent in values & locations.

For our bingo card we shall assume the card is 5x5, the center (free) space is always marked, and the only bingos that count are the traditional single bingo of 5 values marked in a row horizontally, vertically, or diagonally.

Class details:

- Implement the following functions:
    - The default constructor, which should place, in order, 1-5, 16-20, 31-35, 46-50, and 61-65 in the columns.
    - `void fill(...)`: Fills the card.
    - `int how_close()`: Returns the minimum count of values needed to be marked to obtain the bingo on that card.
    - `void mark(int)`: Marks the specific value on the card and displays a message if a bingo is found for the first time only.
    - `void display()`: Displays the card (using `setw(5)` - see example output below) and the marked values (x for marked, - for unmarked).
    - `int same_values(...)`: Returns the count of values such that they are on both cards (but not necessarily in the same location).
    - `bool equals(...)`: Returns true if two cards are identical with respect to both values and locations.
- The ... above indicates that you need to figure out what goes there and/or it may change based on your implementation of members.
- You may use whatever member variables you like for the card and you may have more member functions than the above if you deem it necessary, however use good design choices.

Program details:

- All input will come in the form of a text file that will be given at the command line (e.g. `./a.out data`).
- You are responsible for error checking that a file was given and really exists and display an error message if no filename is provided or the file does not exist.
- You are not responsible for error checking any of the data in the file. All data in the file is guaranteed to be valid.
- Each line in the file will contain one command or data to be used with the previous command. For the `mark` & `fill` commands, the subsequent line will contain the data to be used (1 integer in the case of mark, 25 integers separated by spaces in the case of fill).
- Declare two `bingocard` objects in your main body.
- `f1`, `f2`, `m`, `c`, `d`, `s`, and `e` are the only required commands and are detailed as follows:
    - `f1` or `f2`: fill card 1 or card 2, respectively, with the 25 integers separated by spaces that are in the following line.
    - `m`: on both cards mark the integer found in the following line. If a bingo is obtained (for the first time only) then display a message.
    - `c`: display the minimum number of values needed to be marked to obtain bingo on both cards.
    - `d`: display the card values and marked/unmarked values.
    - `s`: display the count of like values on both cards.
    - `e`: display whether the two cards have the same values & positions.
- For every non-data line in the file, output "`Command: X`" where `X` is the command encountered in the file. For every data line in the file, output "`Data: X`" where `X` is the line of data.
- Upon filling any card (via any method) with values the entire card should be unmarked with the exception of the center (free) space.

Be sure to:

- Make member functions constant as appropriate.
- Make members public or private as appropriate.
- Pass all objects by reference.
- Not use global variables.
- Use proper variable names, spacing, indenting, and commenting per the guidelines.
- Make sure your code compiles according to the guidelines.
- Test your code.
- Try to match the formatting of your output as closely as possible to the example output below to assist in grading.
- Submit your assignment as follows:
    - Your main body should be called `main.cpp`
    - Your class definition header file should be called `bingocard.h`
    - Your class implementation file should be called `bingocard.cpp`
    - These three files, and no others, should be inside a directory called `bingo`
    - `tar` your directory & files up per the guidelines
    - Use the `submit` script with assignment code 11 to submit the `tar` file
- You may submit multiple times electronically, but the last one before the deadline is the only one that will be graded.
- Turn in an identical hard copy. Check the guidelines page for more details.

Example output:

No filename given:

```
$ ./a.out
No file specified!
```

File doesn't exist:

```
$ ./a.out fakefile
Error reading file!
```

Example data file (`data`):

```
d
c
s
e
f1
15 14 13 12 11 30 29 28 27 26 45 44 43 42 41 60 59 58 57 56 75 74 73 72 71
d
c
s
e
m
1
m
17
m
49
d
c
m
65
m
2
m
3
m
4
m
5
d
```

Correct output for `data`:

```
Command: d
Card 1:
    1   16   31   46   61    -    -    -    -    -
    2   17   32   47   62    -    -    -    -    -
    3   18   33   48   63    -    -    x    -    -
    4   19   34   49   64    -    -    -    -    -
    5   20   35   50   65    -    -    -    -    -
Card 2:
    1   16   31   46   61    -    -    -    -    -
    2   17   32   47   62    -    -    -    -    -
    3   18   33   48   63    -    -    x    -    -
    4   19   34   49   64    -    -    -    -    -
    5   20   35   50   65    -    -    -    -    -
Command: c
Card 1 is 4 away and card 2 is 4 away.
Command: s
The cards have 25 values in common.
Command: e
The cards are equal.
Command: f1
Data: 15 14 13 12 11 30 29 28 27 26 45 44 43 42 41 60 59 58 57 56 75 74 73 72
71
Command: d
Card 1:
   15   30   45   60   75    -    -    -    -    -
   14   29   44   59   74    -    -    -    -    -
   13   28   43   58   73    -    -    x    -    -
   12   27   42   57   72    -    -    -    -    -
   11   26   41   56   71    -    -    -    -    -
Card 2:
    1   16   31   46   61    -    -    -    -    -
    2   17   32   47   62    -    -    -    -    -
    3   18   33   48   63    -    -    x    -    -
    4   19   34   49   64    -    -    -    -    -
    5   20   35   50   65    -    -    -    -    -
Command: c
Card 1 is 4 away and card 2 is 4 away.
Command: s
The cards have 0 values in common.
Command: e
The cards are not equal.
Command: m
Data: 1
Command: m
Data: 17
Command: m
```

```
Data: 49
Command: d
Card 1:
   15   30   45   60   75    -    -    -    -    -
   14   29   44   59   74    -    -    -    -    -
   13   28   43   58   73    -    -    x    -    -
   12   27   42   57   72    -    -    -    -    -
   11   26   41   56   71    -    -    -    -    -
Card 2:
    1   16   31   46   61    x    -    -    -    -
    2   17   32   47   62    -    x    -    -    -
    3   18   33   48   63    -    -    x    -    -
    4   19   34   49   64    -    -    -    x    -
    5   20   35   50   65    -    -    -    -    -
Command: c
Card 1 is 4 away and card 2 is 1 away.
Command: m
Data: 65
Bingo! Last number: 65.
Command: m
Data: 2
Command: m
Data: 3
Command: m
Data: 4
Command: m
Data: 5
Command: d
Card 1:
   15   30   45   60   75    -    -    -    -    -
   14   29   44   59   74    -    -    -    -    -
   13   28   43   58   73    -    -    x    -    -
   12   27   42   57   72    -    -    -    -    -
   11   26   41   56   71    -    -    -    -    -
Card 2:
    1   16   31   46   61    x    -    -    -    -
    2   17   32   47   62    x    x    -    -    -
    3   18   33   48   63    x    -    x    -    -
    4   19   34   49   64    x    -    -    x    -
    5   20   35   50   65    x    -    -    -    x
```