• Expand the class CText as follow
new topics are green:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>the containing text</td>
</tr>
<tr>
<td>Constructor</td>
<td>create and initialize a text object</td>
</tr>
<tr>
<td>getWordCount()</td>
<td>returns the number of words in the text</td>
</tr>
<tr>
<td>getWordCount(string word)</td>
<td>returns the number of words passed by parameter &lt;word&gt; use therefore the method find of class string. See documentation: <a href="http://www.cplusplus.com/reference/string/string/find/">www.cplusplus.com/reference/string/string/find/</a></td>
</tr>
<tr>
<td>getLetterCount()</td>
<td>returns the number of letters in the text without spaces, commas and dots.</td>
</tr>
<tr>
<td>getLetterCount(char ch)</td>
<td>counts the number of letters passed by parameter &lt;ch&gt;</td>
</tr>
<tr>
<td>contains(string word)</td>
<td>returns true if &lt;word&gt; is in text, false otherwise</td>
</tr>
<tr>
<td>setFirstLettersToUpper()</td>
<td>changes every first letter of a word to upper case: i.e.: The fox is big. → The Fox Is Big. should even work, if there are more than one spaces between words.</td>
</tr>
<tr>
<td>toString()</td>
<td>for text output of object</td>
</tr>
</tbody>
</table>

Here a main procedure for testing:

```cpp
int main() {
    CText Gedicht("Zum sehen geboren\n, zum Schauen bestellt\n, test test test, etc...");
    cout << "Gedicht: \n" << Gedicht.toString() << endl;
    cout << (Gedicht.contains("gebaren") ? "true": "false") << endl;
    cout << (Gedicht.contains("geboren") ? "true": "false") << endl;
    cout << "Letters: " << Gedicht.getLetterCount() << endl;
    cout << "Words: " << Gedicht.getWordCount() << endl;
    cout << "Number of characters 'e': " << Gedicht.getLetterCount('e') << endl;
    cout << "Number of words 'test': " << Gedicht.getWordCount("test") << endl;
    Gedicht.setFirstLettersToUpper();
    cout << Gedicht.toString() << endl;
    return 0;
}
```

Output:

Gedicht:
Zum sehen geboren, zum Schauen bestellt, test test test, etc...
false
true
Letters: 48
Words: 12
Number of characters 'e': 11
Number of words 'test': 3
Zum Sehen Geboren, Zum Schauen Bestellt, Test Test Test, Etc...
• Expand class CLotto as follow
  new topics are green:

  lottonumbers[]  array that contains the lotto-numbers
  CLotto          Standard-constructor creates numbers in the range of 1
to 49
  CLotto(from,to) overloaded constructor creates numbers in the range
<from> to <to>
generateNewNumbers fill array with lottonumbers
getNumbers      return lottonumbers-array
sort            sort lottonumbers-array ascending, with bubblesort
                algorithm
toString        for text output of object
isDouble        return true if the array <lottonumbers> contains <number>
reset           set all elements of <lottonumbers> to zero

You can test your class with the following main-function:

```cpp
int main()
{
    CLotto numbers(3,77), lotto;

    //numbers from 1 to 49 are drawn:
    cout << "1 to 49:" << endl;
    for(int draw=1; draw<10; draw++) {
        lotto.generateNewNumbers();
        lotto.sort();
        cout << draw << ":" << lotto << endl;
    }

    //numbers from 3 to 77 are drawn:
    cout << 
            \"\n3 to 77:" \" << endl;
    for(int draw=1; draw<10; draw++) {
        numbers.generateNewNumbers();
        numbers.sort();
        cout << draw << ":" << numbers << endl;
    }
    return 0;
}
```

Output:

1 to 49:
1: 3,27,30,31,33,44
2: 4,20,25,36,39,47
3: 7,9,22,24,28,44
4: 7,30,35,36,40,49
5: 4,5,9,21,28,35
6: 2,8,19,24,29,44
7: 7,13,19,40,41,47
8: 1,8,24,27,32,36
9: 1,14,18,25,38,49

3 to 77:
1:10,33,36,47,52,53
2:33,45,58,69,72,73
3:8,11,18,21,37,40
4:3,26,35,51,54,61
5:32,39,48,57,68,74
6:5,9,21,37,51,59
7:5,34,36,44,48,72
8:4,16,23,40,54,57
9:3,7,21,36,68,69
• Please open the file with the class-definition of CVector and do the following extension:

```cpp
class CVector {
    ...
};

ostream& operator<<(ostream& stream, CVector& v) {
    stream << v.toString();
    return stream;
}
```

With this global overload of `<<` it will be possible to output an object of `CVector` without calling explicitly the method `toString()`:

```cpp
CVector v(3,7,-12);
cout << v << endl;  //instead of
cout << v.toString() << endl;
//both kind of output is now possible
```

**Now lets go to the actual task:**

Expand the class `CVector` in a way, that it is possible to write expressions like this:

```cpp
CVector v1(4,-8,-12), v2(7,9,-9);
cout << v1 * v2 << endl;  //does now the same as
cout << v1.crossProduct(v2) << endl;
```

If you need orientation, please look at the example `Fraction` we did in the last semester:


unfold the bar: '22.03.2013 Fraction with overloaded operators and cancel method:'