CSC3002 Assignment 1

Problem 1

Exercise 2.9 and 2.10:

The combinations function $C(n, k)$ described in this chapter determines the number of ways you can choose k values from a set of n elements, ignoring the order of the elements. If the order of the value matters-so that, in the case of the coin example, choosing a quarter first and then a dime is seen as distinct from choosing a dime and then a quarter-you need to use a different function, which computes the number of **permutations**. This function is denoted as $P(n, k)$, and has the following mathematical formulation:

$$P(n, k) = \frac{n!}{(n-k)!}$$

Although this definition is mathematically correct, it is not well suited to implementation in practice because the factorials involved can get much too large to store in an integer variable, even when the answer is small. For example, if you tried to use this formula to calculate the number of ways to select two cards from a standard 52 -card deck, you would end up trying to evaluate the following fraction:

$$\frac{80,658,175,170,943,878,571,660,636,856,403,766,975,289,505,440,883,277,824,000,000,000,000}{30,414,093,201,713,378,043,612,608,166,064,768,844,377,641,568,960,512,000,000,000,000}$$

even though the answer is the much more manageable 2652(52×51).

The $C(n, k)$ function from the text and the $P(n, k)$ function come up often in computational mathematics, particularly in an area called combinatorics, which is concerned with counting the ways objects can be combined.

Write a function **permutations (n, k)** that computes the $P(n, k)$ function without calling the **fact** function in the file **combinatorics.cpp**. When you write the implementation, make sure to rewrite the code for the combinations function so that it uses the efficiency enhancements suggested for permutations.

Requirements: Please finish the file *combinatorics.cpp.* DO NOT modify the main() part, which is for the test unit.

Problem 2

Exercise 3.20:

There is no gene for the human spirit. - Tagline for the 1997 film GATTACA

The genetic code for all living organisms is carried in its DNA-a molecule with the remarkable capacity to replicate its own structure. The DNA molecule itself consists of a long strand of chemical bases wound together with a similar strand in a double helix. DNA's ability to replicate comes from the fact that its four constituent bases-adenosine, cytosine, guanine, and thymine-combine with each other only in the following ways:

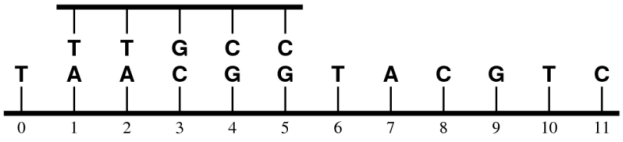
- Cytosine on one strand links only with guanine on the other, and vice versa
- Adenosine links only with thymine, and vice versa.

Biologists abbreviate the names of the bases by writing only the initial letter: **A**, **C**, **G**, or **T**.

Inside the cell, a DNA strand acts as a template to which other DNA strands can attach themselves. As an example, suppose that you have the following DNA strand, in which the position of each base has been numbered as it would be in a C++ string:

Your mission in this exercise is to determine where a shorter DNA strand can attach itself to the longer one. If, for example, you were trying to find a match for the strand

the rules for DNA dictate that this strand can bind to the longer one only at position 1:



By contrast, the strand

matches at either position 2 or position 7.

Write a function

int findDNAMatch (string s1, string $s2$**, int start** $= 0$ **) ;**

that returns the first position at which the DNA strand **s1** can attach to the strand **s2**. As in the **find** method for the **string** class, the optional start parameter indicates the index position at which the search should **start**. If there is no match, **findDNAMatch** should return -1.

Requirements:

Please finish the file *FindDNAMatch.cpp*. DO NOT modify the main() part, which is for the test unit.

Problem 3

Exercise 4.8:

Even though comments are essential for human readers, the compiler simply ignores them. If you are writing a compiler, you therefore need to be able to recognize and eliminate comments that occur in a source file.

Write a function

void removeComments (istream & is, ostream & os)

that copies characters from the input stream **is** to the output stream **os**, except for characters that appear inside C + + comments. Your implementation should recognize both comment conventions:

- Any text beginning with /∗ and ending with ∗/, possibly many lines later.
- Any text beginning with // and extending through the end of the line.

The real C + + compiler needs to check to make sure that these characters are not contained inside quoted strings, but you should feel free to ignore that detail. The problem is tricky enough as it stands.

Requirements:

Please finish the file *RemoveComments.cpp*. DO NOT modify the main() part, which is for the test unit.

Problem 4

Exercise 4.9:

Books were bks and Robin Hood was Rbinhd. Little Goody Two Shoes lost her Os and so did Goldilocks, and the former became a whisper, and the latter sounded like a key jiggled in a lck. It was impossible to read "cockadoodledoo" aloud, and parents gave up reading to their children, and some gave up reading altogether.... - James Thurber, The Wonderful O, 1957

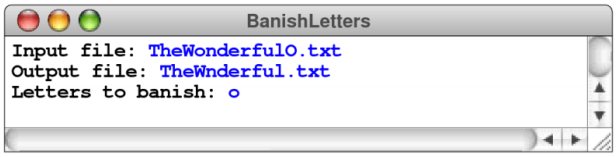
In James Thurber's children's story The Wonderful *O*, the island of Ooroo is invaded by pirates who set out to banish the letter *O* from the alphabet. Such censorship would be much easier with modern technology. Write a program that asks the user for an input file, an output file, and a string of letters to be eliminated. The program should then copy the input file to the output file, deleting any of the letters that appear in the string of censored letters, no matter whether they appear in uppercase or lowercase form.

As an example, suppose that you have a file containing the first few lines of Thurber's novel, as follows:

TheWonderfulO.txt

```
Somewhere a ponderous tower clock slowly
dropped a dozen strokes into the gloom.
Storm clouds rode low along the horizon,
and no moon shown.  Only a melancholy
chorus of frogs broke the soundlessness.
```

If you run your program with the input



it should write the following file:

TheWnderful.txt

```
Smewhere a pnderus twer clck slwly
drpped a dzen strkes int the glm.
Strm cluds rde lw alng the hrizn,
and n mn shwn.  nly a melanchly
chrus f frgs brke the sundlessness.
```

If you try to get greedy and banish all the vowels by entering **aeiou** in response to the prompt, the contents of the output file would be

```
Smwhr  pndrs twr clck slwly
drppd  dzn strks nt th glm.
Strm clds rd lw lng th hrzn,
nd n mn shwn.  nly  mlnchly
chrs f frgs brk th sndlssnss.
```

Requirements:

Please finish the file *BanishLetters.cpp*. The test file *TheWonderfulO.txt* has been provided for your testing(or you can use others files for testing as well). DO NOT modify the main() part, which

is for the test unit.

Requirements for Assignment

Please submit your finished files onto the OJ system. You should finish each .cpp file according to the problem requirements. There would be a pre-test when you submit your code, which contains some test cases, pay attention that the score of pre-test does not mean the final score you obtain as we would have more cases for testing after the due date.

Please note that, the teaching assistant may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself.

Please refer to the BB system for the assignment deadline, late submission will lead to some penalty.

Reminder: Do not submit your code until the late minutes.