San José State University
Department of Computer Science

CS 144
Advanced C++ Programming
Spring 2019
Instructor: Ron Mak

Assignment #3

Assigned: Tuesday, February 12
Due: Tuesday, February 19 at 8:30 AM
URL: http://codecheck.it/files/19021201528vw1cfsi0rtloob95ov2byljx
Canvas: Assignment 3. War and Peace
Points: 100

War and Peace
This assignment will give you practice with C++ strings and vectors.

Download and unzip the complete text of *War and Peace* as an ASCII file from
http://www.cs.sjsu.edu/~mak/CS144/assignments/3/WarAndPeace.txt
The file contains over 65,000 lines and over a half million words. This file has already
been uploaded to CodeCheck.

Write a C++ program to search for the following names in the text:

- Makar Alexeevich
- Joseph Bazdeev
- Boris Drubetskoy

Scan (read and search) the input file a line at a time. For each occurrence of each name
in an input line, print:

- The line number (first line is 1).
- The character position of the first letter of the first name (first position is 1).
- The name itself.

Print the names in line number order. A name can be split across two consecutive lines.
In that case, print the line number of the first name. If you find more than one name in a
line, print the names in alphabetical order.

You may assume that the names in the input file are always in the order of the first name
followed by the last name. Between the first and last names, there is either a single
space or a line break. If a name is split across two lines, the first name will be at the end
of one line, although there may be one or two blanks between the last letter and the line
break, and the last name will always be the first character of the next line.

Strings and vectors
You must use <u>C++ strings</u> and not the old C strings, and you must use <u>vectors</u> instead of arrays. Use the vector **at** method instead of indexing with **[** and **]**.

For string functions, see http://www.cplusplus.com/reference/string/string/
For vector functions, see http://www.cplusplus.com/reference/vector/vector/

Functions
You must have good functional decomposition. Follow the convention of declaring your functions before the main. Include documentation (comments) for each declaration:

- the purpose of the function and any assumptions (preconditions) that it makes
- what each parameter is; use **@param**
- what is the result (postcondition); use **@return** if it is not void

Pass nonscalar parameters by reference to avoid unnecessarily copying argument values. Mark with **const** any parameters whose values the function isn't supposed to change. Define the functions after the main. Document the internals of each function, but do not repeat the documentation of its declaration.

Avoid the use of global variables. Use function parameters instead. Global constants are OK.

Expected output
CodeCheck will check your output for correct values and for formatting.

LINE	POSITION	NAME
19949	1	Boris Drubetskoy
21906	66	Makar Alexeevich
21953	2	Makar Alexeevich
22173	9	Boris Drubetskoy
23910	62	Boris Drubetskoy
23982	39	Boris Drubetskoy
24033	54	Boris Drubetskoy
29439	31	Boris Drubetskoy
29614	28	Boris Drubetskoy
33007	1	Boris Drubetskoy
41110	22	Boris Drubetskoy
46593	60	Joseph Bazdeev
46612	19	Joseph Bazdeev
46622	28	Makar Alexeevich
46626	18	Makar Alexeevich
46626	39	Joseph Bazdeev
46642	20	Joseph Bazdeev
46689	11	Makar Alexeevich
48300	11	Makar Alexeevich
48314	42	Makar Alexeevich
48317	10	Makar Alexeevich
48323	68	Makar Alexeevich
48326	38	Makar Alexeevich
48331	60	Makar Alexeevich
48336	13	Makar Alexeevich
48347	28	Makar Alexeevich
48407	53	Makar Alexeevich
48409	19	Makar Alexeevich
48415	8	Makar Alexeevich
48433	24	Makar Alexeevich
48435	1	Makar Alexeevich
48468	68	Makar Alexeevich

Tips

Write the first version of your program to find only names contained in a single line. Your second version can find names that are split across two lines.

If you search each line for the names in alphabetical order and there are more than name in the line, you will find the names in alphabetical order.

Rubrics

Criteria	Max points
Good output (as determined by CodeCheck) • Correct output values. • Correct output formatting.	**30** • 20 • 10
Good program design • Good use of C++ strings • Good use of vectors. • Good functional decomposition. • Good use of reference and `const` parameters. • Good choice of variable and function names.	**60** • 15 • 15 • 10 • 10 • 10
Good program style • Function declarations before the main with documentation, definitions after the main with internal documentation. • Minimal use of global variables.	**10** • 5 • 5

You can submit as many times as necessary to get satisfactory results, and the number of submissions will not affect your score. When you're done with your program, click the "Download" link at the very bottom of the Report screen to download a signed zip file of your solution.

Submit the signed zip file without renaming it into Canvas:
Assignment #3. War and Peace.

Academic integrity