CS 101 **Project 2** **Summer 2017**

Write a C++ program project2.cpp that behaves as described below. Your program will read from standard input and write to standard output.

First read a sequence of strings until end-of-input occurs; each string occupies an entire line and may include any visible characters as well as blank characters. Next determine which strings are anagrams, and also determine the largest cardinality set of anagrams amongst the given strings. (Here two strings are regarded as anagrams if they are permutations of the same characters; all characters including blanks are significant, and uppercase is distinct from lowercase.) Finally, for each set of anagram strings that has the largest cardinality, display this cardinality followed by the list of anagram strings in lexicographical order.

Your program should behave as illustrated by the following four pairs of example inputs and outputs:

```
SERPENT          3
antler           PRESENT
post             REPENTS
SPOT             SERPENT
rental           3
REPENTS          antler
PRESENT          learnt
learnt           rental
OPTS
Tops
Pots
stop
```

```
nectars          5
despair          canters
medical          nectars
trances          recants
praised          scanter
decimal          trances
recants
aspired
declaim
canters
diapers
claimed
scanter
```

```
- ab             3
A -b             +a B
- bA             B a+
B A-             a +B
-b a             3
B+ A             - bA
bA -             A -b
B a+             bA -
+a b
a +B
+a B
AB +
```

```
all fools' day           4
sabre-toothed tigers'    saber-toothed tiger's
April Fool's day         saber-toothed tigers'
all fool's day           sabre-toothed tiger's
Jews' harp               sabre-toothed tigers'
jews' harp
area-moment method
jew's harp
April Fools' day
saber-toothed tiger's
sabre-toothed tiger's
Jew's harp
moment-area method
saber-toothed tigers'
```

You may assume that the input will contain at most 1,000,000 strings. Both correctness and efficiency are important. You must implement appropriate algorithms and data structures so that your program will run quickly even when the input size is very large.

Your program should be runnable by typing either of the following commands:

./a.out
./a.out < *inputfile* > *outputfile*

For the first line above, your program will read from the keyboard and write to the screen. The second line redirects standard input to come from the specified *inputfile*, and it redirects standard output to go to the specified *outputfile*, but your program is unaware of these files.

Please carefully read the following requirements:

- You must do your own work. You must not borrow any code from any other person, book, website, or any other source. You also must not share your code with any other person, or post it on any website. We run plagiarism detection software on every project. So if you violate these rules, you may receive an invitation to the dean's office to discuss the penalties for academic misconduct.
- Because a purpose of this course is to learn how to properly implement data structures and algorithms, it is not permitted in general to include STL libraries such as <list>, <vector>, <stack>, <queue>, <deque>, <algorithm>, <numeric>, <utility>, … in your program. You are always permitted to use <iostream>, <string>, and any C libraries such as <cstdlib> and <cmath>. But do not include any other libraries unless explicitly allowed on the assignment.
- Make sure your program runs properly on the cs-intro.ua.edu server, because this is where your program will be graded. In particular, make sure your program initializes the values of all variables when they are declared or allocated. Otherwise it might behave differently on Linux than it does on a PC or Mac.
- Your program will be compiled using this command: g++ project2.cpp –Wall –lm –std=c++11. Alternatively, if you split your program into multiple files, then it will instead be compiled using this command: g++ *.cpp –Wall –lm –std=c++11.
- Verify that all the necessary .h and .cpp files which are needed to compile your program are included in the same directory before you submit your program. There should be no extra subdirectories and no extra .h or .cpp files, otherwise your program might not compile the way you intended.
- Compress your project into a zip file that contains your C++ program source file. Right-click (or secondary click) on your project directory, and then (depending on your operating system) select either the Compress option or Send To → Compress from the popup menu. Finally upload your .zip file that contains your .cpp file for this project to Blackboard.
- If you violate the above requirements such that it breaks our grading script, your project will be assessed a significant point deduction, and extreme or multiple violations may cause the project to be considered ungradable.
- Every semester many students lose some points because they don't follow all the instructions. So please read and follow all the project specifications precisely to prevent losing points unnecessarily. If anything is unclear, please ask for clarification well before the project is due. Please pay particular attention to input and output formats.
- Submit your project on Blackboard by the due date (11:59pm Friday). There is a grace period of 24 hours (until 11:59pm Saturday). Projects submitted on Sunday will be assessed

a late penalty of 5% per hour. No projects will be accepted after Sunday. Once it is graded, your project score will be posted on Blackboard and the results of the grading script will be sent to your Crimson email account.

- Double-check and triple-check your submission when you submit it. Errors discovered later cannot be fixed and resubmitted after the project is graded. Projects will not be re-graded unless an error is found in the grading script or in the input/output files that are used during grading.