HW4
Out: October 11, Wednesday -- DUE: October 18, Wednesday, 12:00pm (NOON)
EC327 Introduction to Software Engineering – Fall 2023

Total: 100 points

- *Comment your code and use a clean, easily understandable coding convention. A few coding style guides for your reference:*

http://geosoft.no/development/cppstyle.html
https://google.github.io/styleguide/cppguide.html

Submission: ***Failure to follow submission guidelines WILL result in loss of points on the assignment.*

1. Completed assignments **must** be submitted on GradeScope.
2. Write your answers clearly in your favorite text editor, and submit a .**cpp** file for each question (Q1.cpp, Q2.cpp, Q3.cpp) on GradeScope.
3. Please make sure your code compiles and runs as intended on the **engineering grid**. Code that does not compile on the eng-grid will NOT be graded and will receive a 0.

Late Homework Submissions:
Recall the late submission policy and the fixed penalty of 20% for submitting up to two days after the deadline. Also recall that we only grade your latest submission.

Please write C++ code for solving the following problems. You are strongly encouraged to study relevant chapters in your textbook (e.g., Chapters 8-9 for arrays and pointers).

Q1. ***Arrays, C-strings, pointers.*** **[35 points]**
Write a C++ function that computes the largest common sub-string of two C-strings. Largest common sub-string examples:
Boston and Killington -> ton
James and Thames -> ames
computer and Counter -> ter
abc and abba -> ab
software and MICROSOFT -> none (i.e., comparison is case sensitive)

Your function prototype is as follows:
```
char* largestCommonSubstr(const char s1[ ], const char s2[ ]);
```
The function returns a pointer to the first element of the resulting C-string. If a common sub-string does not exist, then the returned pointer should be pointing to a *null terminator*.

Put all your code for your function implementation in a single file called *Q1.cpp (you may put your function prototype in an .h file while testing but will not be submitting any .h files!).* Make sure to test your function thoroughly, but do not submit your test code. SUBMIT ONLY THE FUNCTION CODE. Including code other than your function implementation code in Q1.cpp WILL result in a loss of points. Please follow requirements exactly as your code will go through an autograder.

Hint: Note that you can compute the length of input C-strings through the functions in C++ C-string library.

Q2. *Pointers and dynamic memory allocation* **[30 points]**
Write a function that triples the size of an array. The function prototype is as follows:
```
int* TripleCapacity(int* list, int size);
```

The function should return a pointer to an integer array that has a size equal to *3*size*. The values in the new array at indices from 0 to size-1 should have the same values as in the original input array in the same order. The rest of the values in the new array should be all initialized to 0.

Make sure to test your function thoroughly, but do not submit your test code. SUBMIT ONLY THE FUNCTION CODE. Including code other than your function implementation code in Q2.cpp WILL result in a loss of points. Please follow requirements exactly as your code will go through an autograder.

Q3. *Arrays* **[35 points]**
Write a C++ function to report the most frequent character in a given character array as follows:

```
char MostFrequentChar(char word[ ], int size) {
//returns the most frequent character in the input char array word
      …
}
```

Implement a "case-insensitive" version of your function; that is 'm' and 'M', 'E' and 'e', etc. should be considered as the same character.

```
Char MostFrequentCharCaseInsensitive(char word[ ], int size) {
/*returns the most frequent character (case-insensitive) in the input char array
word*/
            …
}
```

Hint: the offset between the ASCII values of an uppercase and lowercase version of a letter is always the same (e.g., if the offset between 'A' and 'a' is n, then the offset between 'M' and 'm' is also n).

Put both function implementations in *Q3.cpp (again, you will not be submitting any .h files!)*. Make sure to test your functions thoroughly, but do not submit your test code. SUBMIT ONLY THE FUNCTION CODE. Including code other than your function implementation code in Q3.cpp WILL result in a loss of points.