**CMPE 50 – Fall 2018**

**Lab #4 – C-strings and string class.**

**Exercise 1 (Progr. Proj.: 8.1)**

(*Use C-string to solve this problem.)*

Write a program that reads in a sentence of up to 100 characters and outputs the sentence with spacing corrected and with letters corrected for capitalization. In other words, in the output sentence, all strings of two or more blanks should be compressed to a single blank. The sentence should start with an uppercase letter but should contain no other uppercase letters. Do not worry about proper names; if their first letters are changed to lowercases, that is acceptable. Treat a line break as if it were a blank space, in the sense that a line break and any number of blank spaces are compressed to a single blank space. Assume that the sentence ends with a period and contains no other periods. For example, the input

noW iS thE TiMe fOr aLl

gOOD MEN TO ComE TO tHe

aId

oF

ThE CounTRY.

should produce the following output:

Now is the time for all good men to come to the aid of the country.

**Exercise 2 (Prog Proj 8.7)**

*(Use the string class to solve the problem)*

Write a program that can be used to train the user to use less sexist language by suggesting alternative versions of sentences given by the user. The program will ask for a sentence, read the sentence into a *string* variable, and replace all occurrences of masculine pronouns with gender-neutral pronouns. For example, it will replace “he” with “she or he”, and “him” with “her or him”. Thus, the input sentence

See an adviser, talk to him, and listen to him.

should produce the following suggested changed version of the sentence:

See an adviser, talk to her or him, and listen to her or him.

Be sure to preserve uppercase letters for the first word of the sentence. The pronoun “his” can be replaced by “her(s) or his”; your program need not decide between “her” and “hers”. Allow the user to repeat this for more sentences until the user says she or he is done.

You need to take care of the following string replacements:

he/she -> she or he

him/her -> her or him

his/her(s) -> her(s) or his

## Algorithm/Pseudo-code for Exercise 1

1. This exercise requires you to compress several consecutive “whitespaces” into a single space. A whitespace can be a space ‘ ‘, a tab ‘\t’, or a newline ‘\n’ and/or ‘\r’. So it is convenient to define a function to check if a character is a whitespace or not. For example, define the following function to accomplish it.

bool IsWhitespace(char ch);

1. The beginning whitespaces, if any, need to be removed.
2. Read the input character one at a time, call IsWhiteSpace() to check if it is a whitespace. If not, write the character to the output. If yes, go to 3.a. Repeat this step until all characters are read.
   1. You would need a bool flag to help you identify if the current whitespace is the first one between two words. If so, writes a space ‘ ‘ to the output string; if not, do not write anything.
   2. Reset this flag if the next character is not a whitespace. Go back to step 3 if more characters are to be read in the string.
   3. Special case needs to be handled for the very beginning whitespace of the input string. Use a bool variable to help you.

**bool** is\_firstblank = **false**; // Used for the compressing consecutive whitespaces

**bool** is\_initial = **true**; // Used for removing the beginning whitespaces

// Read in all characters and change all characters to lowercase and compress

// consecutive whitespaces to a single blank.

**for** (**int** i = 0; i < str length; i++)

{

**if** (IsWhitespace(str[i]))

{

**if** (is\_initial)

{

// Skip the beginning whitespaces

**continue**;

}

// compress all consecutive whitespace into one blank

**if** (is\_firstblank == **false**)

{

add a space to the new string

is\_firstblank = **true**;

}

} **else**

{

**if** (is\_initial)

{

// We have already got the first non-whitespace character

// So there are no more beginning whitespaces

is\_initial = **false**;

}

is\_firstblank = **false**;

add the tolower(character) to the new string

}

}

## Algorithm/Pseudo-code for Exercise 2

1. This exercise is more complicated than it looks. You cannot just use a string variable and use >> to read a string at a time since there are punctuation marks, such as comma ‘,’ or period ‘.’, that you don’t want to be included in a string. So you will have to read in a character at a time and decide whether it is an alphabet or not. So it is convenient that you define the following function:

bool IsAlphabet(char ch);

1. Read in one character at a time, check if it is an alphabet, if so, concatenate it to the current word, using a string “+” operator, for example. When you finish building a word, you can then work on the replacement as in the next step.

string word = "";

// Read a word containing alphabets only

**while**(**true**)

{

cin.get(next\_symbol);

**if** (is\_alphabet(next\_symbol))

{

word = word + next\_symbol;

} **else**

{

cin.putback(next\_symbol);

**break**;

}

};

1. Define a replacement function by checking the input string to be one of the “sexist” words. If so, change the word to be the a-sexist version. Remember to handle if the word is capitalized or not.