

# SMALL PROGRAM 7

---

COP3223C Introduction to Programming with C  
Dr. Andrew Steinberg

Fall 2022

## Due Date

The assignment is due on November 14th at 11:59pm EST via Webcourses. **Do not email the professor or TAs your submissions as they will not be accepted!** This assignment is accepted late up to 24 hours with a penalty. Please see the syllabus for more information on this. Make sure to submit on time to get potential full credit. Make sure to also take into consideration the uploading time. In the past, students who are working last minute on the assignment sometimes run into uploading issues where their Internet may run slow, resulting in late submissions. The timestamp Webcourses uses for your submission will be applied and will be the final say. Please do not email the instructor or TAs saying your Internet was running slow. If the time is off by a second of the due date, then the assignment is considered late. Plan accordingly!

## Important! Read Carefully!

This assignment contains a set of problems that are to be completed in **one C file**. You have learned about creating user-defined functions and why they are so beneficial to us programmers. For each problem in the assignment, you will create the definition of the user-defined function that is asked in the description. **If you do not create a user-defined function for each of the problems, then you will receive no credit for the problem.** Creating user-defined functions is good practice! You also must write the function prototypes! Missing function prototypes will result in points being deducted. Function prototypes are also good practice as well. The file must be named *smallprogram7\_lastname\_firstname.c*, where lastname and firstname is your last and first name (as registered in webcourses). For example Dr. Steinberg's file would be named *smallprogram7\_Steinberg\_Andrew.c*. Make sure to include the underscore character `_`. If your file is not named properly, points will be deducted. The script the graders use will pull your name from the file name properly. It is imperative you follow these steps so you can get your points!

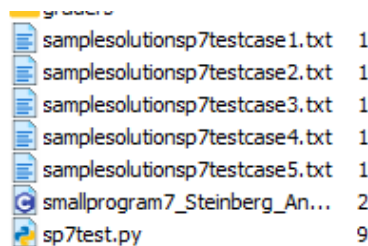
## Testing on Eustis

It is your responsibility to test your code on Eustis. If you submit your assignment without testing on Eustis, you risk points being deducted for things that may behave differently on your operating system. Remember, you cannot dispute grades if your code didn't work properly on

Eustis all because it worked on your machine. The Eustis environment gives the final say. Plan accordingly to test on Eustis!!

## The Python Script File! Read Carefully!

A python script has been provided for you to test your code with a sample output of Dr. Steinberg's solution. This script will check to make sure your output matches exactly with Dr. Steinberg's solution file as the graders are using this to grade your assignments. The script removes leading and trailing white space, but not white space in the actual text. If there is anything off with the output (including the expected answer), the script will say your output is not correct. This includes your output producing the correct answer, however there is something off with the output display. The script is going to run 5 unique scenarios for each problem (5 Test Cases). Each test case contains a different set of input values being used to ensure your code produces the correct answer. Back in your previous assignments, Dr. Steinberg would provide 1 sample solution that you would upload to Eustis. Now, there are 5 solution text files you are going to need to upload to Eustis. Before you test your program, your directory in Eustis should look something like this: If you have these files, you are ready to run the script. Use the following



samplesolutionsp7testcase1.txt	1
samplesolutionsp7testcase2.txt	1
samplesolutionsp7testcase3.txt	1
samplesolutionsp7testcase4.txt	1
samplesolutionsp7testcase5.txt	1
smallprogram7_Steinberg_An...	2
sp7test.py	9

Figure 1: Your setup for testing on Eustis. 5 sample txt files (provided for you in Webcourses), your C program, and the python test script.

command to test your code with Dr. Steinberg's provided solution sample.

---

```
python3 sp7test.py
```

---

After you run the script, 5 new text files are going to be generated. These files are the solution output for each test case. If the script says your output is incorrect, checkout the sample text file that was generated (a new text file will be created from the script that contains YOUR output). If your numbers are off or different from Dr. Steinberg's, then that means there is something not right with code's logic and calculating the answer. However if your numbers match with Dr. Steinberg's solution, then that means there is extra/missing white space or newlines detected. Compare the text file generated by the script with solution text file line by line to find the missing/extra white space or newlines. Once you believe you found the error, rerun the script to see if the output matches.

samplesolutionsp7testcase1.txt	1
samplesolutionsp7testcase2.txt	1
samplesolutionsp7testcase3.txt	1
samplesolutionsp7testcase4.txt	1
samplesolutionsp7testcase5.txt	1
smallprogram7	16
smallprogram7_Steinberg_An...	2
sp7student_output_test_cas...	1
sp7student_output_test_cas...	1
sp7student_output_test_cas...	1
sp7student_output_test_cas...	1
sp7student_output_test_cas...	1
sp7test.py	9

Figure 2: Your Eustis setup after running the script in Eustis.

## The Rubric

Please see the assignment page for the established rubric on webcourses.

## Comment Header

Make sure you place a comment header at the top of your C file. You will use single line comments to write your name, professor, course, and assignment. For example, Dr. Steinberg's header would be:

---

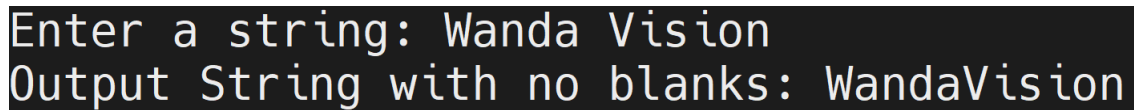
```
//Andrew Steinberg
//Dr. Steinberg
//COP3223C Section 1
//Small Program 7
```

---

Missing a comment header will result in point deductions!

## Problem 1

Write a user defined function called `deblank` that takes a string phrase output and a string phrase input argument and copies the input argument with all blanks removed and stores the result in the output string. Blanks are the white spaces in a string. For this problem, declare the strings of size 20 and ask the user to input one to store. Display the output result in the main function. The following figure shows a sample output in the terminal.



```
Enter a string: Wanda Vision
Output String with no blanks: WandaVision
```

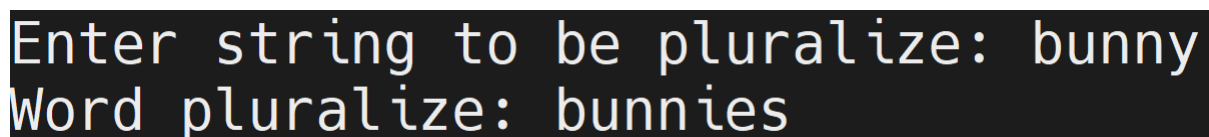
Figure 3: Sample output for problem 1. Make sure it matches for the python script.

## Problem 2

Write a user defined function called `pluralize` that takes a noun as a string input and will pluralize the word. The result should be displayed in the main function. Note the following rules about noun plurals:

- a) If noun ends in “y”, remove the “y” and add “ies”.
- b) If noun ends in “s”, “ch”, or “sh”, add “es”.
- c) In all other cases, just add “s”.

For this problem, declare a string of size 20 and ask the user to input one. The following figure shows a sample output in the terminal.



```
Enter string to be pluralize: bunny
Word pluralize: bunnies
```

Figure 4: Sample output for problem 2. Make sure it matches for the python script.

### Problem 3

Write a user defined function called `fact` that takes a one word string as input and determines how many letters, digits, and punctuation characters it contains. The results should be displayed in the user defined function. For this problem, declare a string of size 20 and ask the user to input one. The following figure shows a sample output in the terminal.

```
Enter a string with no spaces: ab12c,3?ad.!!x0
ab12c,3?ad.!!x0 has
7 letters
3 digits
5 punctuation characters
```

Figure 5: Sample output for problem 3. Make sure it matches for the python script.

### Problem 4

Write a user-defined function called `substring` that takes two strings as arguments. If the second string (in the argument) is contained, the memory address at which the contained string begins is returned. For example, `substring("turkey", "key")` would return the memory address of the `k` in "turkey". Otherwise return a NULL pointer. In the main function, ask the user for the two strings. Once the strings are entered, call the user defined function. Once the function has returned a resulting value, display message of the result. For example `substring("turkey", "key")` would cause the message "Substring key exists!" to be displayed. The function call `substring("store", "too")` would cause the message "Substring too doesn't exist!" to be displayed. Note: The strings are only single words. Do not worry about sentences! **You cannot use the built in `strstr()` function to solve this problem or any built in memory function! Using any of these functions will result in a score of 0 for this question. You can assume that the first string argument is always bigger than the second string argument.** For this problem, declare a string of size 20 and ask the user to input one. The following figure shows a sample output in the terminal.

**Super Big Hint:** Your function is going to return an address type value, think about how your user defined function header would look. You should have notice something during lectures.

```
Enter the string: turkey
Enter the substring you would like to look for: key
Substring key exists!

Enter the string: store
Enter the substring you would like to look for: too
Substring too doesn't exist!
```

Figure 6: Two sample outputs for problem 4 when the substring exists and doesn't exists. Make sure it matches for the python script.