

CS 135 Fall 2010: Project 3 Due December 5, 11:59pm.

The goal of this project is to further expose you to working with pointers, strings, memory allocation, working with functions and file I/O operations. It builds on your knowledge of data structures – be sure to read the example on linked lists in the textbook (Chapter 19) very carefully.

- You must work on your own on this project – no collaboration of any kind allowed, and you **CANNOT** use source code from any other source apart from the textbook or lecture notes.

In this project you will write a program to create a string, and later manipulate it, of ASCII characters by reading them from a file. Specifically, you are given a text file - “input_project3.txt”. This file contains 200 characters (without any separator between them). These characters fall in the range of 33-125 of ASCII values. This input file will be randomly generated whenever the “main” function gets executed (this functionality is already present in the main function; students don’t need to implement it).

A sample of input_project3.txt would look like this:

jVnh2#[xTh4%]#ILASz6QM89tl.Q':IqFg6%hZ1,:KMS\$YED8&9K!4QOn^Cy

You are provided 6 blank C files for each function that must be implemented. You are allowed to add functions/files if needed, but you CANNOT choose to implement all functions in one single C file. You are also provided a main.c file that contains the main function.

You begin by reading the input file character by character (e.g., using “getchar”). For every character read, you create a separate node and attach it to the linked list.

You are required to implement the following functionalities

1. Adding/Attaching a node to the end of linked-list/string. Perform this task in “add_node.c” file.
2. Counting the length of the string. This is a trivial task. However, this exercise should prepare your appetite for more complex tasks. Implement this task in “count_length.c” file.
3. Counting the types of different characters in the string. The characters in the input file will be of three types - lower case, upper case and special characters (\$, # , % etc.). Your program must print the total count of each type. Implement in “count_types.c”.
4. Check if a character is present in the string. This task must be implemented in “if_present.c” file. This function will be required by the following requirement.
5. Delete all duplicates in the string. This task must be implemented in “delete_duplicate.c” file. It is absolutely necessary that you FREE the memory for the node that is deleted. Besides, you must make sure that deleting a node doesn’t break the string into multiple strings. You must print the string on console after you delete all the duplicates.

6. Reverse the order of the string. Implement this task in “reverse_order.c” file. Here is it mandatory to reverse the pointers - not merely exchange characters. You will not earn any grade for this task if you simply swap the characters.

Note: You must FREE the memory for deleted nodes as well as the entire linked-list at the end of the all tasks. You will lose points if you do not do so.

Remarks: It should be amply clear to you that global variables will serve no purpose here. Besides, please do not email the TA if you get compilation errors. With gcc the location and line number of errors is shown very clearly. Also, although you are not allowed to use somebody else’s source code, you are free to make Google/Bing search for getting help on segmentation faults and compilation errors only!

A sample output for this project would look like this

```
Length of the string is: 200
Count, Lower Case: 100
Count, Upper Case: 50
Count, Special Characters: 50
String after deleting duplicates: !n%6^3erTY#
Reverse order:#YTre3^6%n!
```

Other Requirements

1. You must print the output to a file - “output_project3.txt”. This must be implemented using FILE operation and not by redirecting the output to a file from the terminal.
2. You must turn in your Makefile (a small sample makefile called “sample_make” will be provided).
3. The name of the executable must be “project3_lastname.o”

Grading and Submission Instructions:

- You must turn in, using blackboard, a tar (or zip) file containing (1) a short document (report) describing your implementation, (2) the source code and (3) the makefile. Please DO NOT turn in 7z extension zip files.
- Your project report must state the name of team member clearly on the top.
- Your project report must also specify the implementation platform - Cygwin/Linux/Mac OS/Hobbes(solaris). If you have implemented the project in Visual Studio, then it is your responsibility to test and make it run on Cygwin.
- The project is worth 40 points towards the total projects grade. You will be graded on both correctness (50%) as well as efficiency (35%) of your solution, in addition to documentation and code style (15%).
- Your code must compile correctly – if you submit code that does not compile, you will receive a zero on the project.

Please clarify any questions you have (regarding project requirements) with the TA.