

# Lab 4

## Strings and vectors

*30 September 2024*

The first program in this week's lab will be on codeboard, and it will read a single string—specifically, it will read one string that doesn't have spaces in it—and print out information about it. Remember to test (submit) your program after each step!

1. First, write the program to read the string and print out the message

```
The word _____ has _ characters.
```

except with the blanks filled in correctly for that string: the string itself and its length (how many characters it has).

2. Add second line to the output that says

```
Its first is '_' and its last is '_'.
```

filling in the blanks with the first and last characters of the string (you can assume it will not be empty).

3. In cases where the string is more than two characters long (and only in those cases), add a third line

```
If you trim the first and last characters it leaves ____.
```

The blank here should have the whole string *except* for its first and last characters.

4. Finally, print one line that says

```
The characters are:
```

and then lines with the characters of the word, in order (each on its own line).

Don't forget to include documentation comments at the top of your program. Attend carefully to punctuation and spelling!

I'll be circulating around the lab to answer questions. If you're stuck on some aspect of the first part, ask me about that (and while you're waiting

for me to get to you, look over the next section). If you're not stuck but haven't finished the first part, work on that now. If you're done with the first part, continue on to the next section.

## Part 2: Print initials

The purpose of this whole lab is to give practice with strings and vectors, so we don't have interesting word problems to work through—all the parts of this lab have a sort of disconnected flavour about them (sorry).

In this part, you'll write a program that processes a vector of strings and prints out the initial character in each string. For instance, if the vector contained the strings "Longwood" and "University", the program's output would be "LU". Use an accumulator to build the eventual output inside the loop, and then print it after the loop.

You may assume none of the strings are the empty string. (You could skip it if you encounter it, but I'm not checking for this.) I do expect that you'll try running your program with different values in the vector, but since the only way we have (for now) of making vectors with values is to put them in when the vector is defined in `main`, there's no separate place to define test cases; you're on your own for making sure it's well-tested and correct.

Do this problem on codeboard, in the Lab 4 Part 2 assignment. (Regarding "frequent handin", this one's pretty short, but still aim for at least one intermediate handin, maybe right after defining the vector of strings.)

## Part 3: Transform

You'll do this part on the department server, so create a directory for this lab if you haven't already; don't forget to put a `readme` file in there, and remember that you'll have to write some test cases (`.in` and `.expect`) and show in the `readme` how to use them to test your code.

In this part, you'll write a program that reads in a single word (which might include some punctuation) and then prints it back except with every punctuation character replaced with a period and everything else printed in uppercase. Use an accumulator variable to build the string that will eventually be printed, and then actually print it after the loop is done.

There are a few possible ways you might identify the different kinds of characters. As with other parts of this lab, remember to re-read the course pack for what you know about strings and characters (including some things we haven't explicitly discussed during lecture), and some of the various computations you can do with them!

## AI policy and frequent submission

(no change from the Lab 3 version of this policy)

Some use of generative AI is fine, but you a) should not paste this assignment or type it verbatim into the AI prompt, and b) should not be asking the AI for the whole program all at once. (Just like you can ask for help from a human, but should not have them write the whole program for you!) If you get help from an AI chat OR from a person, you should note that in a program comment near whatever you got from them.

Relatedly, I expect that you'll click the Submit button/run the `handin` program relatively often, and I *require* that you do so at least 2–3 times over the course of working on the lab. As a rule of thumb, hand it in after completing each 1–2 points on the rubric. Submissions that jump straight to a final, (near-)correct version with no intervening submissions along the way *will receive little or no credit* for that part. ← scoring note!

This is a new policy that I'm experimenting with; let me know if you have any feedback.

## Handing in

It's due as usual on Monday at 4pm. The first two parts are already on codeboard and should be submitted via the "submit" button there; Part 3 needs to be handed in on the server. Hand in using the same command as before (`handin`), this time with assignment name `lab4`.

## Rubric

### RUBRIC (Tentative)

- 1 Attendance at lab with preview done or question written down
- 1 Appropriate documentation in each part
- string info**
- <sup>1</sup>/<sub>2</sub> Program compiles and runs, reads string, prints it out
- 1 Uses `.at`, `.length` correctly
- 1 Uses `if` and `.substr` correctly
- 1 Uses `for` to print individual letters
- Print initials**
- 1 Define and init a vector of string (with valid strings)
- 1 Accumulator concatenates initials inside `for`, prints after loop
- Transform**
- <sup>1</sup>/<sub>2</sub> Program compiles and runs, reads string, prints something
- 1 Accumulator concatenates transform inside `for`, prints after loop
- 1 Test cases, good coverage \*

\* To get the full point for good test case coverage, you have to EITHER pass all test cases in the group OR indicate in the readme which ones aren't passing. This is your way of showing me that you've actually run your tests.

*This document was written and prepared without the use of generative AI.*