CS 137: Assignment #7

Due on Friday, Nov 14, 2025, at 11:59PM

Submit all programs using the Marmoset Submission and Testing Server located at
https://marmoset.student.cs.uwaterloo.ca/

Victoria Sakhnini

Fall 2025

Notes:

- Use the examples to guide your formatting for your output. Remember to terminate your output with a newline character.
- For this assignment, you may use any content covered until the end of Module 10.
- `<math.h>` is not allowed
- Use Valgrind when testing.

Problem 1

For this problem, you will write a function `void lookBehind(const char* check, const char* s_sub, char* word);` that removes all occurrences of `s_sub` from `word` preceded by `check`. Note that you will apply the same process to all updated versions of `word` that occur due to removing `s_sub`.

You are to complete and submit `lookbehind.c` file containing only your implemented function (you must delete the test cases portion/the `main` function). However, **you must keep the required included libraries, and any helper functions you defined.**

The sample code for testing is in the provided C file.

Problem 2

You are tasked with creating a program that reads a string of uppercase characters and spaces and performs a series of complex cipher operations. The input string consists of a sentence containing only uppercase letters and spaces. Your goal is to implement a C program, `cipher.c`, that reads the input string, applies the following operations, and prints the final result:

Reverse: Reverse the entire string.

Replace: Replace each vowel (A, E, I, O, U) with the next consonant (non-vowel) in the alphabet. If the vowel is 'U', replace it with 'B'.

Shift: Replace each consonant(not space) with a new character located to the right by a number of positions equal to its ASCII value modulo 10 (if you reach Z you continue with A. For example, B will replaced with H because the ASCII of B is 66. 66 module 10 is 6. H is the Character located 6 positions to the right in the Ascii table. Another Example: X will be replaced with F because the ASCII of X is 88, 88 module 10 is 8, after X comes Y, then Z, then A,B,C,D,E,F.

Write the C program `cipher.c` to perform the described operations on the input string and print the final result. Assume that the input string has at most 100 characters. You must submit the file containing your implemented functions **and** the `main` function, **you must keep the required included libraries, and any helper functions you defined.**

The headers of the required functions are provided in the sample test below:

```c
#include <stdio.h>
#include <string.h>

// Function to reverse the entire string
void reverseString(char *str) {

}

// Function to replace each vowel with the next consonant
void replaceVowels(char *str) {

}

// Function to shift each consonant to the right by a number of positions
// equal to its ASCII value modulo 10
void shiftConsonants(char *str) {

}
// DO NOT CHANGE MAIN
int main(void) {
    char inputString[101];

    // Read the input string
    printf("Enter a string (uppercase letters and spaces only): ");

   scanf("%100[^\n]s", inputString);

    // Reverse the entire string
    reverseString(inputString);
    printf("Reversed String: %s\n", inputString);

```

```
    // Replace vowels with the next consonant
    replaceVowels(inputString);
    printf("Replaced Vowels String: %s\n", inputString);

    // Shift consonants to the right
    shiftConsonants(inputString);
    // Print the final result
    printf("Ciphered String: %s\n", inputString);

    return 0;
}

```

Sample input:

Enter a string (uppercase letters and spaces only): I LOVE SOFTWARE ENGINEERING

Expected output:

Reversed String: GNIREENIGNE ERAWTFOS EVOL I

Replaced Vowels String: GNJRFFNJGNF FRBWTFPS FVPL J

Ciphered String: HVNTFFVNHVF FTHDXFPV FBPR N

Sample input:

Enter a string (uppercase letters and spaces only): I

Expected output:

Reversed String: I

Replaced Vowels String: J

Ciphered String: N

Sample input:

Enter a string (uppercase letters and spaces only): SE COURSES

Expected output:

Reversed String: SESRUOC ES

Replaced Vowels String: SFSRBPC FS

Ciphered String: VFVTHPJ FV

Sample input:

Enter a string (uppercase letters and spaces only): I LOVE CS ONE THREE SEVEN

Expected output:

Reversed String: NEVES EERHT ENO SC EVOL I

Replaced Vowels String: NFVFS FFRHT FNP SC FVPL J

Ciphered String: VFBFV FFTJX FVP VJ FBPR N

Problem 3

In this question, we will simulate lossless string compression for a string of characters where the compression method involves encoding-decoding as follows:

Consecutive occurrences of the same data value are stored as a single occurrence of that data value and a count of its consecutive occurrences rather than the original run.

Encoding "AAAABBBBBBBBBBCCdDAA" results in "A4B11C2d1D1A2"

Assume that the original string to code includes only alphabetical letters and that the string to decode is in the correct format.

Implement the functions:

```
char* compressString(const char* str);
```

```
char* decompressString(const char* str)
```

You are to complete and submit `rlestring.c` file containing only your implemented functions (you must delete the test cases portion/the `main` function). However, **you must keep the required included libraries, and any helper functions you defined.**

Sample main:

```
int main() {
    char* compressedStr = compressString("ABBBBBTUuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuGHHHKHHHKK");
    printf("Compressed String: %s\n", compressedStr);
    char* decompressedStr = decompressString(compressedStr);
    printf("Decompressed String: %s\n", decompressedStr);
    free(compressedStr);
    free(decompressedStr);
    return 0;
}
```

Expected output:

Compressed String: A1B5T1U1u39G1H3K1H3K2

Decompressed String: ABBBBBTUuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuGHHHKHHHKK

Problem 4

You are given a C program that defines functions to search for characters in a C-style string (const char *). The functions utilize various techniques to locate specific characters or groups of characters within a string. Your task is to implement similar functions with the following specifications and then test them to ensure they work correctly.

Problem Description

Implement the following functions:

1. **const char* find(const char *p, int n, char c)**
   - Searches for the first occurrence of character c in the range [p, p + n).
   - Returns a pointer to the location of c if found, or NULL if c is not in the specified range.
2. **int find_first_of(const char *p, int pos, const char *q, int count)**
   - Searches for the first character in the range [p + pos, p + strlen(p)) that is also in the range [q, q + count).
   - Returns the position of the first matching character in p if found, or -1 if no match is found.
3. **Int find_first_not_of(const char *p, int pos, const char *q, int count)**
   - Searches for the first character in [p + pos, p + strlen(p)) that is not in [q, q + count).
   - Returns the position of the first character in p that does not match any character in q, or -1 if all characters match.
4. **int find_last_of(const char *p, int pos, const char *q, int count)**
   - Searches for the last character in [p, p + pos] that matches any character in [q, q + count).
   - Returns the position of the last matching character in p if found, or -1 if no match is found.
5. **int find_last_not_of(const char *p, int pos, const char *q, int count)**
   - Searches for the last character in [p, p + pos] that does not match any character in [q, q + count).
   - Returns the position of the last character in p that does not match any character in q, or -1 if all characters match.
6. **int mystrlen(const char *p)**
   - returns the length of the string (similar to strlen in <string.h>

Note: You must not include <string.h>

You are to submit `strlib.c` file containing only your implemented functions (you must delete the test cases portion/the `main` function). However, **you must keep the required included libraries, and any helper functions you defined.**

You are provided the main function (in strlib.c file) to help you with testing.