CS 137: Assignment #7

Due on Friday, Nov 15, 2024, at 11:59PM

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

Victoria Sakhnini

Fall 2024

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 below.

```
int main(void){

    char word[] = "foo bar loop hoop" ;
    lookBehind("l", "oo", word);
    assert(0==strcmp("foo bar lp hoop", word));
    lookBehind("o", "o", word);
    assert(0==strcmp("fo bar lp hop", word));

    char word3[] = "CS137137" ;
    lookBehind("CS", "137", word3);
    assert(0==strcmp("CS", word3));
    char word3b[] = "CS137 137" ;
    lookBehind("CS", "137", word3b);
    assert(0==strcmp("CS 137", word3b));

    char word4[] = "111111" ;
    lookBehind("1", "1", word4);
    assert(0==strcmp("1", word4));

    char word5[] = "wow" ;
    lookBehind("", "w", word5);
    assert(0==strcmp("o", word5));

    return 0;
}

```

Problem 2

Write a function `char* merge (char* s1, char* s2)` that takes in two strings containing words of alphabetical characters `(A-Z,a-z)` separated by a single space and returns a pointer to the result of merging the two sentences allocated on the heap. The result must not contain extra spaces—only one space between every two words.

For instance, if `s1="The brown jumps the dog"`, `s2="quick fox over lazy"`, then the returned string should be the string `"The quick brown fox jumps over the lazy dog"`.

The two input sentences need not be so balanced regarding the number of words. For example, if `s1="the brown"`, and `s2="quick fox is sleeping today"`, the expected returned string should be `"the quick brown fox is sleeping today"`. This is because the first three words we get by merging are `"the quick brown"`, but then `s1` is depleted by this point, so we shall append the remainder of `s2` to our output string.

You may assume the first and last characters of the input string are NOT spaces. Assume that each sentence has at least one word in it.

You are to submit `mergestrings.c` file containing only your implemented function (that is, you must delete the test cases portion/the `main` function). However, **you must keep the required included libraries.**

The sample code for testing is below.

```c
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <string.h>
int main(void)
{
    char s1[] = "The brown jumps the dog";
    char s2[] = "quick fox over lazy";

    char *s = merge(s1, s2);
    assert(!strcmp(s, "The quick brown fox jumps over the lazy dog"));
    free(s);

    char s3[] = "the brown";
    char s4[] = "quick fox is sleeping today";
    s = merge(s3,s4);
    assert(!strcmp(s, "the quick brown fox is sleeping today"));
    free(s);

    char* s5 = "happy to you";
    char* s6 = "birthday";
    s = merge(s5,s6);
    assert(!strcmp(s, "happy birthday to you"));
    free(s);

    return 0;
}
```

Problem 3

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, **<u>you must keep the required included libraries, and any helper functions you defined.</u>**

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: FVTHPJ 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
```