CS 137: Assignment #7

Due on Friday, Nov 18, 2022, at 11:59PM

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

Victoria Sakhnini

Fall 2022

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 11.
- <math.h> is not allowed
- Use Valgrind when testing.

Problem 1

Create a C program named `semidrome.c` that consists of a function

`bool is_semidrome(char *s)`

which determines whether or not a string `s` is a semidrome, that is, a concatenation of palindromes of length at least 2 (notice that palindromes of length at least 2 are themselves semidromes).

You are to submit this 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.

```
#include <stdbool.h>
#include <string.h>
#include <assert.h>

bool is_semidrome(char *s);

int main(void)
{
         assert(is_semidrome("popeye"));
         assert(is_semidrome("racecar"));
         assert(!is_semidrome("aab"));
          assert(!is_semidrome(""));
         return 0;
 }

```

Problem 2

Write a function `int shortestRepeatingPrefix(char s[])` that takes in a string `s`, and finds the shortest length prefix `p` of `s`, so that `s` is simply `p` repeated multiple times. Instead of returning the prefix as a string, return the number of characters in this prefix of `s`.

For instance, if `s="abcabcabcabc"`, the desired prefix is `"abc"`, because `s` is `"abc"` repeated 4 times. This is shorter than the also valid prefixes `"abcabc"` and `"abcabcabcabc"`. The function should return 3, which is the `"abc"` string length.

If `s="abcdefg"`, then the shortest such prefix is the whole string itself, so return 7.

You are to submit `srp.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 <string.h>
int main()
{
    char s1 [] = "hihihihi";
    assert(shortestRepeatingPrefix(s1) == 2);

    char s2 [] = "aaaaa";
    assert(shortestRepeatingPrefix(s2) == 1);

    char s3 [] = "qwerty";
    assert(shortestRepeatingPrefix(s3) == 6);

    char s4 [] = "";
    assert(shortestRepeatingPrefix(s4) == 0);

    return 0;
}

```

Problem 3

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;
}
```