

CMPT125, Spring 2023

Homework Assignment 2

Due date: Friday, March 5, 2023, 23:59

You need to implement the functions in **assignment2.c**.
Submit only the **assignment2.c** file to CourSys.

Solve all 5 problems in the assignment.

The assignment will be graded automatically.

Make sure that your code compiles without warnings/errors, and returns the required output.

Your code MUST compile in CSIL with the Makefile provided.

If the code does not compile in CSIL, the grade on the assignment is 0 (zero).

Even if you can't solve a problem, make sure the file compiles properly.

Warning during compilation will reduce points.

More importantly, they indicate that something is probably wrong with the code.

Memory leak during execution of your code will reduce points.

Check that all memory used for intermediate calculations are freed properly.

Your code must be readable, and have reasonable documentation, but not too much.

No need to explain `i+=2` with `// increase i by 2`.

An example of a test file is included.

Your code will be tested using the provided tests as well as additional tests.

Do not hard-code any results produced by the functions as we will have additional tests.

You are strongly encouraged to write more tests to check your solution is correct, but you don't have to submit them.

1. You need to implement the functions in **assignment2.c**.
2. If necessary, you may add helper functions to the assignment2.c file.
3. You should not add `main()` to assignment2.c, because it will interfere with `main()` in the test file.
4. Submit only the **assignment2.c** file to CourSys.

Question 1 [15 points].

Write a function that gets 2 positive ints each between 1 and 9999, and outputs the number obtained by their concatenation.

```
long concat_numbers(int n1, int n2)
```

For example:

```
concat_ints(1, 25) should return 125.  
concat_ints(123, 4567) should return 123456.  
concat_ints(999, 9999) should return 9999999.  
concat_ints(9, 8765) should return 98765.
```

You may assume that the input is always legal, i.e., both numbers are between 1 and 9999.

Question 2 [15 points].

Write a function that gets a string and checks if it is a palindrome of even length.

```
bool is_even_palindrome(const char* str);
```

For example:

- On input "a2z@dd@z2a" the function returns true.
- On input "ae11&edwdccek" the function returns false, because it is not a palindrome.
- On input "abcba" the function returns false, as it is not a palindrome of even length.

Problem 3 [20 points]

Write a function that gets a string, and returns the length of the longest substring that is a palindrome of even length.

```
int longest_even_subpalindrome(const char* str);
```

For example:

- On input "-122221a3dcba" the function returns 6.
- On input "abracabra" the function returns 3.
- On input "abcd" the function returns 2.
- On input "abcdefedcba@cda**k2**2k**bcd" the function returns 6.

For full marks your solution needs to work on inputs of length up to 10,000 under one second.

Question 4 [30 points].

Write a function that gets two strings containing positive integers, and outputs a new string containing their sum.

```
char* add(const char* num1, const char* num2)
```

For example, `add("123456789", "987654321")` returns `"1111111110"`.

1. You may assume that the input is always legal, i.e., both strings are positive numbers
2. Note that the numbers may be larger than the maximum of `int` or `long`.
3. Also make sure that the returned string is created on the heap (and not as a local variable).

Question 5 [20 points].

Write a function that gets a string `num` containing a positive integer, and outputs a new string containing $2 \times \text{num}$.

```
char* mult2(const char* num)
```

For example, `mult2("1234706")` returns `"2469412"`.

1. You may assume that the input is always legal, i.e., the string is a positive integer
2. Note that the numbers may be larger than the maximum of `int` or `long`.
3. Also make sure that the returned string is created on the heap (and not as a local variable).