CMPT125, Fall 2022

Homework Assignment 1
Due date: Friday, September 30, 2022, 23:59

You need to implement the functions in ***assignment1.c***.
Submit only the ***assignment1.c*** file to CourSys.

Solve all 5 problems in this assignment, one function for each problem.

Grading: The assignment will be graded automatically.
Make sure that your code compiles without warnings/errors, and returns the required output.

Compilation: 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.

Warnings: Warnings during compilation will reduce points.
More importantly, they indicate that something is probably wrong with the code.

Dynamically allocated arrays: Do not use variable length arrays! Never!
If you need an array of unknown length, you need to use malloc.

Memory leaks: Memory leaks during execution of your code will reduce points.
Make sure all memory used for intermediate calculations are freed properly.

Readability: Your code must be readable, and have reasonable documentation, but not too much. No need to explain i+=2 with // increase i by 2.
Write helper functions if that makes the code more readable.

Testing: 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 all the functions in ***assignment1.c***.
2. You should not add main() to assignment1.c, because it will interfere with main() in the test file.
3. Submit only the ***assignment1.c*** file to CourSys.

Problem 1 [10 points]
Write a function that gets two integers `x` and `y`, and returns the number of even integers between `x` and `y`, including them.

```
long count_even(long x, long y);
```

For example:

- `count_even(1, 5)` *returns 2 (because the even numbers are 2 and 4).*
- `count_even(7, -8)` *returns 8 (the even numbers are -8,-6,-4,-2,0,2,4,6).*
- `count_even(6, 6)` *returns 1 (the only even number is 6).*
- `count_even(-11, -11)` *returns 0.*

Problem 2 [20 points]
Write a function that gets an array of ints of length n>0, and returns the most frequent element in the array. If there is more than one such element, the function may return any of them.

```
int most_frequent(const int* arr, int n);
```

For example:

- *On input* `[1,7,-3,7,4,7]` *the function returns* `7` *because it appears 3 three times.*
- *On input* `[-2,-3,-2,-3]` *the function may return either* `-2` *or* `-3`.
- *On input* `[111]` *the function returns* `111` *because 111 appears once.*

Problem 3 [10 points]
Write a function that gets a string and checks if it is a palindrome of odd length.

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

For example:

- *On input* `"a2z@d@z2a"` *the function returns* `true`.
- *On input* `"ae11&edwdccek"` *the function returns* `false`, *because it is not a palindrome.*
- *On input* `"abccba"` *the function returns* `false`, *as it is not a palindrome of* ***odd*** *length.*

Problem 4 [20 points]
Write a function that gets a string, and returns the length of the longest substring that is a palindrome of odd length.

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

For example:

- *On input* `"-12121a3dcba"` *the function* `5`.
- *On input* `"abracadabra"` *the function* `3`.
- *On input* `"abba"` *the function* `1`.
- *On input* `"abcda@12bcdak2m_m2kbcd"` *the function* `7`.

For full marks your solution needs to work on inputs of length up to 100,000 under one second.

Problem 5 [40 points]

Write a function that gets a string representing a positive integer and a digit between 1 and 9. The function divides the number by the digit, and returns the string representing the result of the division with remainder. The format of the answer needs to be `"result`**`R`**`remainder"`*. That is, the string should contain a number, followed by letter* `"R"` *followed by the remainder (the remainder should be one digit long).*

```
char* str_div_by_digit(const char* num, int digit);
```

For example:

- `str_div_by_digit("15", 6)` *returns* `"2R3"`.
- `str_div_by_digit("5", 9)` *returns* `"0R5"`.
- `str_div_by_digit("918273", 1)` *returns* `"918273R0"`.
- `str_div_by_digit("0", 4)` *returns* `"0R0"`.
- `str_div_by_digit("123765712931258731249371253 9", 7)` *returns* `"176808161330369616070530362R5"`.

Notes:

1. *You may assume that the input is always legal, i.e., the string is a positive integer correctly formatted, and there are no unnecessary leading zeros, etc, and 1<=digit<=9.*
2. *Note that the numbers may be so large that they do not fit into* `int` *or* `long`*. That is, you should not try to convert string to int.*
3. *The remainder might be zero (see examples above).*
4. *Remember to use malloc appropriately and return the string allocated on the heap (and not be a local variable).*