CMPT125, Spring 2022

Homework Assignment 1
Due date: Friday, February 4, 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 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 ***assignment1.c***.
2. If necessary, you may add helper functions to the assignment1.c file.
3. You should not add main() to assignment1.c, because it will interfere with main() in the test file.
4. Submit only the ***assignment1.c*** file to CourSys.

Question 1 [15 points].
Write a function that gets two ints `x` and `y`, and returns the sum of all ints between `x` and `y`, including them.

```
int sum_interval(int x, int y);
```

For example:

- `sum_interval(1, 4)` *returns* `1+2+3+4=10`.
- `sum_interval(10, 3)` *returns* `3+4+5+6+7+8+9+10=52`.
- `sum_interval(2, -1)` *returns* `(-1)+0+1+2=2`.
- `sum_interval(-1, -1)` *returns* `-1`.

1. *Note that we may have `x>=y` or `x<=y`.*
2. You may assume that the sum will be within the range of int.

Question 2 [15 points].
Write a function that gets a string `str`, changes all lower case letters to `*` (asterisk), and all upper case letters to `$` (dollar sign). All other characters remain unchanged. The function returns the total number of letters modified.

```
int hide_letters(char* str);
```

For example:

- *If `str` is "`12ab0`", then the function changes it to "`12**0`", and returns 2.*
- *If `str` is "`aBCDe*`", then the function changes it to "`*$$$**`", and returns 5.*
- *If `str` is "`0123&***`", then the function keeps the string as is, and returns 0.*
- *An empty string is also a string.*

Question 3 [20 points].
Implement the function that gets an array of ints of length n, and returns the number of times the maximal element appears in the array. You may assume that n>0.

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

For example:

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

Question 4 [35 points].
Write a function that gets a string representing a positive integer and an int between 0 and 9. The function multiplies that two numbers and returns the result in a string.

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

For example:

- `mult_number_by_digit("12340", 6)` *returns* `"74040"`.
- `mult_number_by_digit("9", 0)` *returns* `"0"`.
- `mult_number_by_digit("8", 1)` *returns* `"8"`.
- `mult_number_by_digit("9999999999999", 3)` *returns* `"29999999999997"`.

1. *You may assume that the input is always legal, i.e., the string is a positive integer correctly formatted, and 0 <= digit <= 9. Assume there are no unnecessary leading zeros, etc.*
2. *Note that the numbers may be so large that they do not fit into 4 bytes or 8 bytes. That is, you should not try to convert string to int.*
3. *Remember to use malloc appropriately and return the string allocated on the heap.*

Question 5 [15 points].
Write a function that gets an array of n ints, where each entry in the array is between 0 and 9. The function multiplies all these numbers and outputs the result as a string. The reason it needs to output a string (and not an int) is because the result might be too large to fit into an int/long. You may assume that n>0.

```
char* mult_digits(const int* digits, int n);
```

For example:

- `mult_digits([3], 1)` *returns* `"3"`.
- `mult_digits([9,9], 2)` *returns* `"81"`.
- `mult_digits([5,5,4,5,5], 5)` *returns* `"2500"`.
- `mult_digits([1,2,2,2,1,2,4,4,2,2], 10)` *returns* `"1024"`.
- `mult_digits([9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9], 20)` *returns* `"12157665459056928801"`.

1. *You may assume that the input is always legal, i.e., 0 <= digit[i] <= 9 for all i=0…n-1.*
2. *Remember to use malloc appropriately and return the string allocated on the heap.*
3. *If you are allocating memory for some intermediate calculations, remember to free it before returning to avoid memory leaks.*