CMPT125, Spring 2026

Homework Assignment 1
Due date: Friday, January 30, 2026, 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 provided Makefile.
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!
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.
        Submit only the ***assignment1.c*** file to CourSys.

Question 1 [15 points]
Write a function that gets two ints `a` *and* `b`.
If `a>b` *the function returns* a^3+b^2, *and otherwise it returns* a^2+b^3.

```
int square_cube(int a, int b);
```

For example:
- `square_cube(1, 2)` *should return* 1+8=9.
- `square_cube(10, 3)` *should return* 1000+9=1009.
- `square_cube(2, -1)` *should return* 8+1=9.
- `square_cube(-2, -1)` *should return* 4-1=3.

Question 2 [15 points]
Write a function that gets 3 pointers `int* a, int* b, int* c`, *and rotates the values in their addresses to the left. That is,* `a` *gets the value of* `b`, `b` *gets the value of* `c`, *and* `c` *gets the value of* `a`.

```
void rotate3(int* a, int* b, int* c);
```

For example,
- *if we have* `int x=1, y=2, z=3`, *then after calling* `rotate3(&x, &y, &z)` *we should have* `x==2, y==3,` *and* `z==1`.
- *if we have* `int x=7, y=1, z=6`, *then after calling* `rotate3(&x, &y, &z)` *we should have* `x==1, y==6,` *and* `z==7`.

Question 3 [20 points]
Implement the function that gets a string `str`, *changes all digits of* `str` *to* 0 *(zero), and returns the number of digits in the string.*

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

For example:
- *If* `str` *is* `"12ab0"`, *then the function should change it to* `"00ab0"`, *and return 3.*
- *If* `str` *is* `"hello world"`, *then the function should keep it as is, and return 0.*

[Hint 1: you can check if a char is a digit using its numerical value. The numerical values of the digits are consecutive. For example, we have `'3'` *+ 2 ==* `'5'`*]*

Question 4 [20 points]

*Implement the function that gets an array of ints and its length, and returns the maximum of the **absolute values**.*

```
int max_abs(const int* arr, int len);
```

For example:
- *On input* `[1,-3,7,4]` *the function should return* `7`.
- *On input* `[2,-3,2,-4]` *the function should return* `4`.
- *On input* `[0,-3,2]` *the function should return* `3`.

Question 5 [30 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. 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 the returned string is allocated on the heap.*