

CMPT125, Fall 2025

Homework Assignment 1

Due date: Friday, September 26, 2025, 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 floats x and y , and returns the sum of all integers between x and y . If x and/or y are integers, the sum will include them.

```
int sum_ints_interval(float x, float y);
```

For example:

- `sum_ints_interval(0.4, 4.1)` returns `1+2+3+4=10`.
- `sum_ints_interval(7.1, 3.5)` returns `4+5+6+7=22`.
- `sum_ints_interval(4.89, 4.1)` returns `0`.
- `sum_ints_interval(2, -1)` returns `(-1)+0+1+2=2`.
- `sum_ints_interval(-1, -1)` returns `-1`.

1. Note that we may have $x \geq y$ or $x \leq y$.
2. You may assume that the sum will be within the range of `int`.

Problem 2 [10 points]

Implement the function `hide_digits` that gets a string `str`, changes all digits of `str` to `*`, and returns the number of characters in the string that were modified.

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

For example:

- If `str` is `"abC1231"`, then the function change it to `"abC****"`, and returns `4`.
- If `str` is `"a2z21"`, then the function change it to `"a*z**"`, and returns `3`.
- If `str` is `"w**aK*"`, then the function does not modify the string, and returns `0`.

Problem 3 [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 4 [20 points]

Write a function that gets 3 pointers `int* a`, `int* b`, `int* c`, and sorts the values in their addresses in the non-decreasing order. When the function returns, the values in the corresponding addresses should satisfy $*a \leq *b \leq *c$.

```
void sort3(int* a, int* b, int* c)
```

For example, if we have the variables `int x = 10`, `y = 2`, `z = 6`, then after the call `sort3(&x, &y, &z)`, we will have `x = 2`, `y = 6`, and `z = 10`.

Problem 5 [40 points]

Write a function that gets two strings containing non-negative integers. The function adds the two integers, and returns a string containing the sum.

```
char* str_sum(const char* num1, const char* num2);
```

For example:

- `str_sum("2", "4")` returns "6".
- `str_sum("0", "159")` returns "159".
- `str_sum("12", "15")` returns "27".
- `str_sum("999", "2")` returns "1001".
- `str_sum("123456789", "987654321")` returns "1111111110"

1. You may assume that the input is always legal, i.e., the strings are positive integers.
2. Note that the numbers may be larger than the maximum of `int` or `long`. 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 (and a local variable).