CS 137: Assignment #5

Due on Friday, Oct 31, 2025, at 11:59PM

Submit all programs using the Marmoset Submission and Testing Server located at

https://marmoset.student.cs.uwaterloo.ca/

Victoria Sakhnini

Fall 2025

Notes:

- Use the examples to guide your formatting for your output. Remember to terminate your output with a newline character.
- For this assignment, you may use any content covered until the end of Module 8.
- You should use `double` when appropriate; do not use `float` type.
- **Starting A5: The final grade will be 80% correctness (Marmoset results) and 20% style (MarkUs results).**

Problem 1

Write a C program `grades.c` that reads the grades of a specific course and provides some statistics as illustrated in the examples below. Please note the following:

- Your program must read grades using `scanf` until a character is entered.
- Assume that all entered grades are between 0 – 100, inclusive. (This is a critical point to be able to solve the problem without using VLA)
- We also provided `string.txt` to avoid any typos in formatting the outputs. **Make sure to use it for printing**.
- Assume that the user will enter at least two valid grades.
- Failing grade is a grade <50.
- **You must NOT use a variable length array (VLA). Using VLA will result in receiving a zero on this question.**
- **You may not limit the maximum number of grades that can be entered. Failing to do so will result in receiving a zero on this question.**

The **sample standard deviation** formula looks like this:

Formula	Explanation
$s = \sqrt{\frac{\sum (X - \bar{x})^2}{n - 1}}$	• s = sample standard deviation • $\sum$ = sum of... • X = each value • $\bar{x}$ = sample mean • n = number of values in the sample

$$\text{mean} = \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Sample input 1:

```
80
78
80
96
85
78
85
85
9
44
45
66
45
13
?
```

Sample output 1:

```
The maximal grade is: 96
The number of students who received the maximal grade is 1
The minimal grade is: 9
The number of students who received the minimal grade is 1
The course mean is: 63.500
The standard deviation is: 27.734
The number of students who failed the course is: 5
```

Sample input 2:

```
100
100
100
50
30
30
30
30
0
0
0
*
```

Sample output 2:

```
The maximal grade is: 100
The number of students who received the maximal grade is 3
The minimal grade is: 0
The number of students who received the minimal grade is 3
The course mean is: 42.727
The standard deviation is: 40.023
The number of students who failed the course is: 7
```

Problem 2

You are given a dataset of employees' information, which includes their ids, ages, and salaries. The dataset is represented using an array of structures named Employee. Your task is to implement a C program that performs the following operations (some of them are already implemented for you in the provided `company.c` file):

Input: Read data for n employees from the user (0<n ≤ 100).

Display: Display the information of all employees.

Average Salary: Calculate and return the average salary of all employees.

Youngest Employee: Find and display the details of the youngest employee. Assume there is only one.

Salary Range: Return the number of employees whose salaries fall within a given range (inclusive).

Salary Raise: update the salary of all employees with a given raise.

Complete `company.c` program that accomplishes these tasks using the provided array of structures and appropriate functions.

Note: You are to submit `company.c` containing only your implemented functions along with the structure definition (that is, you must delete the provided functions, test cases portion and the `main` function). However, **you must keep the required included libraries.**

Here is an example of the execution of the provided main function:

The bold part is the user input

Enter the number of employees (up to 100): **3**
Enter the details of 3 employees:
Employee 1:
ID: **1111**
Age: **45**
Salary: **120500**
Employee 2:
ID: **22222**
Age: **33**
Salary: **145900**
Employee 3:
ID: **3333**
Age: **55**
Salary: **345889**
Employee Details:
Employee 1:
ID: 1111
Age: 45
Salary: 120500.00
Employee 2:
ID: 22222
Age: 33

Salary: 145900.00
Employee 3:
ID: 3333
Age: 55
Salary: 345889.00
Average Salary: 204096.33
Youngest Employee:
ID: 22222
Age: 33
Salary: 145900.00
Enter the salary range:
Minimum Salary: **200000**
Maximum Salary: **300000**
Number of employees in the given salary range: 0
Salaries after 0.05 raise
Employee Details:
Employee 1:
ID: 1111
Age: 45
Salary: 126525.00
Employee 2:
ID: 22222
Age: 33
Salary: 153195.00
Employee 3:
ID: 3333
Age: 55
Salary: 363183.45

Problem 3

Write a C program `ln.c` that approximates the natural logarithm of a given positive number using the Taylor series expansion. The program should use the `math.h` library for comparison and handling double precision for calculations.

The natural logarithm of a number (x) can be approximated using :

$$\ln(x) = \ln\left(\frac{x}{e^k}\right) + k$$

$$k = \lfloor \log_2(x) \rfloor$$

Taylor series approximation of ln(y+1) = y - y^2/2 + y^3/3 - y^4/4 + ...
Where $y+1=x / e^k$

The program must:

- read a positive double number (x) ((x > 0)).
- approximate the natural logarithm of (x) using the first 30 terms of the series.
- compare the result with the value obtained using the log function from the math.h library.
- print both the approximated value and the value from the log function, along with the absolute difference between them, following the format shown below and using the strings provided in lntrings.txt

Example1:
Enter a positive number: 3
Approximated ln(3.00) = 1.09861229
math.h ln(3.00) = 1.09861229
Absolute difference = 0.00000000

Example2:
Enter a positive number: 34
Approximated ln(34.00) = 3.52640085
math.h ln(34.00) = 3.52636052
Absolute difference = 0.00004032

Example3:
Enter a positive number: -3
Invalid input

Problem 4

Complete implementing the Sequence ADT in `sequence.c`. Read carefully through the global constant and function declarations in `sequence.h` to get a better understanding of what each function should do and how it should behave.

You are also provided with a sample program for testing called `test_sequence.c`

Make sure you add more tests to test your implementations.

Note: You are to submit `sequence.zip` containing only `sequence.h` and `sequence.c`

If you implement the functions correctly, the provided `test_sequence.c` file must print the following:

```
[1]
[empty]
[1]
[1,1]
[empty]
[-11]
[-11,-11]
[22,-11,-11]
[3,22,-11,-11]
[0,3,22,-11,-11]
[0,3,22,5,-11,-11]
[0,3,22,5,-11,-11,0,3,22,5,-11,-11]
[0,3,22,5,-11]
```