# CSE 102 Homework Assignment 4

## DUE

December 11, 2020, 23:55

## Description

- This is an individual assignment. Please do not collaborate
- If you think that this document does not clearly describe the assignment, ask questions before its too late.
- This assignment is about simple file read/write, array usage and string operations. You can use pointers and functions. You cannot use dynamical memory allocation. You can use global variables. You can use standard string library operations.

This is a `C` Programming assignment. You will write a `C` program according to the following description.

- Your program will read two input files:
  - values.txt
  - polynomial.txt
- Your program will create a file:
  - evaluations.txt
- Your program will evaluate the same polynomial for each value read from values.txt and write the results to evaluations.txt

### values.txt

This file holds double numbers separated by whitespace.

```
12.5 5 67.89 -6 -13.37
```

There may be as many as `100` double numbers in this file.

### polynomial.txt

This file holds a polynomial in a character array form.

```
5x+23.5x^3-x^2
```

There will only be one polynomial expression. monomials are not ordered according to the powers of the variable x. Monomials can appear multiple times. Example: `5x+ 4.5x^2 - 4.3x`. Here, there are two first order monomials: `5x` and `-4.3x`. The coefficient of `x` at each monomial is written before the character `x`. Powers of `x` is represented by character `^` followed by a number. Except the constant, each monomial will certainly include a character `x`. There may be more than one constant monomial. Example: `3x+ 3.2 - x^4 + 5.3 + 12.5x^2 - 13.04`. In this example, there are `3` constant monomials: `3.2`, `5.3` and `-13.04`. Test your code with any sort of polinomial you can create according to this description. Examples:

- `x.`
- `x^2.`
- `5.`
- `5+5.`
- `x^20+5.`
-

The length of a polynomial expression can reach up to `1000` characters. This includes spaces. There may be spaces between `+`, `-`, and monomial characters. Example:

- `x       + 3x^3-       5 + x^7.`

There may be spaces between any character. Example:

- `x + 3     x  ^ 3 -    5 + x^     7.`

### evaluations.txt

This file will hold the results of polynomial evaluations for each value read from `values.txt`. If your polynomial string is `5x+23.5x^3-x^2`, set `x` to the value(one of the numbers read from values.txt) and evaluate the mathematical expression: `evaluation = 5*x + 23.5*x*x*x - x*x`. For the given example above, evaluations.txt will be as follows:

```
45804.69
2937.50
7349081.25
-5142.00
-56410.13
```

## Remarks:

- In order to convert char arrays to numbers, you can use function `sscanf()` which is defined in `<stdio.h>`. For example:

```
double d1,d2;
char a[] = "12.5 63.4"
sscanf(a, "%lf%lf", &d1, &d2);
/* d1 stores 12.5 and d2 stores 63.4 */
```

- In order to find powers of a number, you can use `pow()` function defined in `<math.h>`. Be careful with linking your program when you include `math.h`. Some versions of `GCC` will require you to explicitly link the required library.
- You don't have to do error checking on the input file. You can safely assume that you will be given a proper input file which doesn't violate the described format.
- Be careful with the size of the array you allocate in program stack. Large arrays may not fit in program stack(stack size may be smaller on the test machine) and your program crashes.
- Make sure you can read input files with or without a tailing newline at the end. (If you are using a windows machine, newline is `CRLF`, on unix it is `LF`). You can alter this using advanced editors (i.e. Visual Studio Code). Test your code for every possible combination.
- Do not print anything other than the expected output. (For this assignment, your program prints NOTHING).
- You cannot use anything which is not covered in class.
- Do not submit any of the files you used for testing.
- Do not submit your output file.

## Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_<id>.c`.
- Example: `gokhan_kaya_000000.c`. Please do not use any Turkish special characters.
- You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.
- Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used.
- Make sure you don't get compile errors when you issue this command : `gcc <full_name>_<id>.c`.
- A script will be used in order to check the correctness of your results. So, be careful not to violate the expected output format.
- Provide comments unless you are not interested in partial credit. (If I cannot easily understand your design, you may loose points.)
- You may not get full credit if your implementation contradicts with the statements in this document.

## Late Submission

- Late submission is NOT accepted.

## Grading (Tentative)

- `Max Grade` : 100.

All of the followings are possible deductions from `Max Grade`.

- No submission: `-100`. (be consistent in doing this and your overall grade will converge to `N/A`) (To be specific: if you miss `3` assignments you'll get `N/A`)
- Compile errors: `-100`.
- Irrelevant code: `-100`.
- Major parts are missing: `-100`.
- Unnecessarily long code: `-30`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values).
- Significant number of compiler warnings: `-10`.
- Not commented enough: `-5`. (Comments are in English).
- Source code encoding is not `UTF-8` and characters are not properly displayed: `-5`. (You can use 'Visual Studio Code', 'Sublime Text', 'Atom' etc... Check the character encoding of your text editor and set it to `UTF-8` ).
- Fails at reading the file: `-30`.
- Fails during file write: `-30`.
- Wrong evaluation for a single input value: `-10`.
- Wrong evaluation for `2` or more input values: `-40`.
- Wrong evaluation for every input value: `-80`.
- Output format is wrong: `-30`. (Be careful with spacing)
- Infinite loop: `-90`.
- Prints anything extra: `-30`.
- Unwanted chars and spaces in `evaluations.txt`: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.
- IF YOU DON'T FOLLOW THE FILE NAMING CONVENTIONS YOU WILL GET `0`.

Note: Some of these items are not independent. So, you cannot expect isolation of many of them. For example, if you cannot read input file correctly, you will fail to produce the correct output file. Partial grading is not guaranteed.