CSE 102 Programming Assignment 3

DUE

November 26, 2021, 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.

You won't be given a chance to correct any mistakes.

You are going to write a complete C program which implements the following functionality:

- 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. The coefficient of x at each monomial is written before the character `x`. Powers of x is represented by character `^` followed by a number. Each monomial will certainly include a character `x`.

The length of a polynomial expression can reach up to `1000` characters.

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 be 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:

- First degree monomial will not have `^` character in it. Example: `6x`.
- If the coefficinet is `1`, it is not written. Example: `x^2`, `x`, `x^12`.
- There won't be a constant. Smallest possible degree is `1`.

- 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>`

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 that your program does not require specific encodings/markings/line-ending-chars. Make sure it works with a file created in a linux environment.
- 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

- Not accepted.

Grading (Tentative)

- `Max Grade : 100`.
- Multiple tests(at least `5`) will be performed.

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

- hard-coded values `-10`.
- 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`.
- inefficient implementation: `-20`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values, avoid hard-to-follow expressions, avoid code repetition, avoid unnecessary loops).
- 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`).
- Missing or wrong output values: `Fails the test`.

- Output format is wrong: `-30`.
- Infinite loop: `Fails the test`.
- Segmentation fault: `Fails the test`.
- Fails 5 or more random tests: `-100`.
- Fails the test: `deduction up to 20`.
- Prints anything extra: `-30`.
- Requires space/newline at the end of the file: `-20`.
- Requires specific newline marking (CR/LF): `-20`.
- Unwanted chars and spaces in output: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.