CSE 102 Programming Assignment 3

DUE

November 10, 2024, 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`
  - `polynomials.txt`
- Your program will create a file:
  - `evaluations.txt`
- Your program will evaluate the **multiple** polynomials 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 3
-6 -13.37
```

There may be as many as `200` double numbers in this file.

Each line is a pair of values for (x,y) variables of the polynomials.

polynomials.txt

This file holds a polynomial at **each line** in a character array form. **Polynomials are independent**. (Each line will be evaluated independently)

```
y-   5xy^3+   23.5y^2x^3-x^2
10x^2+      x+        xy
3x^3
```

Monomials are not ordered according to the powers of the variable x and/or y. The coefficient of a monomial is written before the character `x` and/or `y`. Powers of `x` or `y` is represented by character `^` followed by a number(small positive integer). Each monomial will certainly include a character `x` and/or `y`. (There will not be a constant term.)

The length of a polynomial expression can reach up to `1000` characters. (max length of a line is `1000` excluding the end-of-line chars.) **There is no limit on the number of polynomials**. You have to read the file *line-by-line* till the end of file. Don't try to save all the lines in an array at once. Process one line at a time. **Polynomials are independent**

evaluations.txt

This file will hold the results of each polynomial evaluations for each successive pairs of value read from `values.txt`. (The first number will the x value, the second number will be the y value.)

Polynomials are independent.

Evaluations of the first polynomial is in the first line. Evaluations of the second polynomial is in the second line etc...

Evaluate every polynomial for all of the value pairs in `values.txt`.

Print only two digits after the decimal point

Evaluation values are separated with single space characters

Remarks:

- **If you repeatedly parse the same information from file or from array, you will get `0`.** You have to parse a polynomial, store the representation. Use the same representation for multiple evaluations. Discard the representation. You have to repeat this procedure for all of the polynomials. (One polynomial at a time.) (You don't have to remember all of the polynomials all at the same time. They are independent)
- **If you try to store all of the `polynomials.txt` in an array, you will get `0`.**
- First degree monomial will not have `^` character in it. Example: `6x` or `7yx^3`
- If the coefficient is `1`, it is not written. Example: `x^2`, `xy`, `y^5x^12`.
- There won't be a constant. Smallest possible degree is `1`.
- The variables `x` and `y` are not ordered. `y` appear before `x` or vice versa.
- The coefficients can be floating point numbers.
- Degree values are positive integers.
- Monomials will not repeat. Example: if there is a monomial `3xy^4`, there won't be another monomial with the same degree values. `xy^4` will be unique. But, there may be a `x^4y` monomial.
- The polynomial expression will never start with a `-` character or `+` character.
- The numbers (coefficients, degree values) may be multiple digit values.
- 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>_PA3.c`.
- Example: `guthrie_govan_PA3.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>_PA3.c`. State the required link options as a comment at the beginning of your source file.
- 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 will be performed.

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

- hard-coded values `-10`.
- No submission: `-100`.
- 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`.