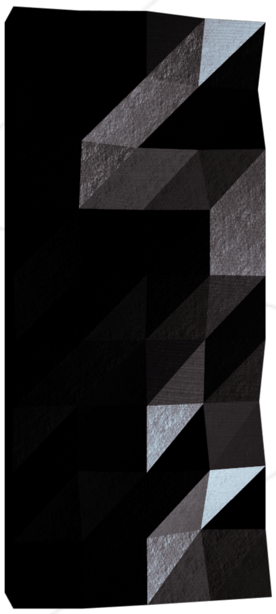

C Bootcamp

Mini-project 02 : evalexpr

Staff WeThinkCode_ info@wethinkcode.co.za

Summary: Third mini-project of the C Bootcamp @ WeThinkCode_.

Contents

Chapter I

Foreword

`TO BE REPLACED`

Voici ce que `Wikipedia` a à dire sur Pinkie Pie :

```
Pinkie Pie est une pouliche à la robe rose avec une crinière et une queue
magenta. Sa marque de beauté se compose de trois ballons. Son caractère est
celui d'un poney enjoué et blagueur, bavard, un petit peu brusque et très
imprévisible. Elle a toujours le dernier mot et ne se décourage jamais. C'est
elle qui organise les fêtes de Ponyville et elle s'occupe à mi-temps d'une
pâtisserie-confiserie. Bien que toutes les protagonistes chantent
occasionnellement au fil de la série, c'est le plus souvent Pinkie Pie qui
choisit de s'exprimer en musique. Pinkie Pie est proches des personnages de
cartoon comiques traditionnels, apparaissant par le haut de l'écran, comptant
sur 6 sabots, faisant de la corde à sauter sans personne pour tenir la corde
ou encore en apparaissant avec des objets sorties de nul part tel que le Party
Canon. Elle habitait a une ferme aux rochers quand elle était poulaine.
Elle représente le rire dans les Éléments de l'harmonie.
```

N'oubliez pas que `sharing kindness, it's an easy feat`.

Chapter II

Consignes

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as you can be.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called `Norminator` to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass `Norminator`'s check.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- If ft_putchar() is an authorized function, we will compile your code with our `ft_putchar.c`.
- You'll only have to submit a main() function if we ask for a program.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.

- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses `gcc`.
- If your program doesn't compile, you'll get `0`.
- Exercises in shell have to be executed with /bin/sh.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask the neighbor on your right. Otherwise, try the neighbor on your left.
- Your reference guide is called `Google / man / the Internet / ....`
- Try checking out the "C Bootcamp" part of the forum on the intranet.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor ! Use your brain !!!

Chapter III

Subjet

	Exercise 00
eval_expr	
Turn-in directory : *ex*00/	
Files to turn in : `Makefile, and all of your program's files`	
Allowed functions : `write, malloc, free`	
Notes : `n/a`	

- Create a program called `eval_expr`.
- It'll have a function `eval_expr` prototyped as follows :

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

- This function takes a characters string as argument. This string represents an arithmetic expression. For example :

  `"3 + 42 * (1 - 2 / (3 + 4) - 1 % 21) + 1"`

- This expression must be calculated and its result must be returned.
- The string passed as argument will be valid (no bugs, no bogus addresses, no letters or syntax errors, no division by zero, etc...).
- The following five operators must be supported :
  - `+` for addition
  - `-` for subtraction
  - `/` for division

  - `*` for multiplication
  - `%` for modulo

- The function must also support any amount of brackets.
- Here's your `main` :

```
int main(int ac, char **av)
{
  if (ac > 1)
  {
    ft_putnbr(eval_expr(av[1]));
    ft_putchar('\n');
  }
  return (0);
}
```