## Интерпретатор

Разглеждаме учебния език EXPR. Той се състои от два оператора: оператор за присвояване и оператор за извеждане на данни. Програмите на този език са редици от оператори за присвояване или извеждане. На всеки ред на една програма на EXPR може да има точно един от двата типа оператори. Имената на променливите в нашия език са съставени само от малки латински букви. Освен това езикът разполага с цели положителни числа в интервала unsigned long int. Операторът за присвояване позволява на дадена променлива да се присвои или число, или стойност на прост аритметичен израз. Съответно операторът за извеждане на резултата позволява отпечатването на стойността на променлива или на число, като отпечатва освен десетичния запис на числото и съответния му двоичен, ограден в кръгли скобки. Всеки ред от нашия език, може да бъде описан със следната контекстно-свободна граматика Г = <N, T, Line, P>, чийто продукционни правила имат вида:

**Line -> Var = Num | Var = Expr | print Var | print Num**
**Var -> a | b | ... | z | aVar | bVar | … zVar**
**Num -> 0 | ... | 9 | 1Num | ... | 9Num**
**Expr -> Expr + Term | Expr - Term | Term**
**Term -> Term * Factor | Term / Factor | Term % Factor | Factor**
**Factor -> Var | Num | (Expr)**

Обърнете внимание, че аритметичните изрази, с които разполага нашият език, допускат употребата на скоби.

Целта на тази задача е да се напише интерпретатор на езика EXPR. Входно/изходните изисквания към програмата са следните. Чете от входен текстов файл програмата на EXPR, изпълнява така написаните оператори и извежда резултата на стандартния изход. Ако интерпретаторът срещне оператор за извеждане на стойност, чиято стойност е недефинирана – например променливата няма присвоена стойност, се извежда подходящо съобщение. Също така операторът за присвояване да предупреждава за деление на нула. Освен това, ако интерпретаторът срещне ред, който не е валиден оператор или израз на езика EXPR, да предупреждава по подходящ начин – например чрез съобщението ‘Syntax Error at line #’.

Тъй като нашият език разполага само с цели положителни числа, то за операцията деление се подразбира целочислено деление.

Пример (входна програма записана във файла p01.txt):
a = 1\r\n
b = 2\r\n
c = 3\r\n
d = a + 2*b%c\r\n

```
print a\r\n
print b\r\n
print c\r\n
print d\r\n
```

Изход:
1 (1)
2 (10)
3 (11)
2 (10)

**Насоки**

Една идея е изразите и операторите да се представят по един и същи начин – чрез двоични дървета. Възлите в тези дървета ще са от няколко типа:

- Цяло число - няма наследници;
- Променлива – няма наследници;
- Оператор за присвояване (ляв наследник - името на променливата, десен наследник - число или дърво на аритметичния израз, чиято стойност трябва да се присвои на променливата);
- Оператор за извеждане (ляв наследник, съдържащ името на променливата, която ще се извежда).

От тук насетне на всеки ред от нашата програма съпоставяме двоично дърво. Задачата се свежда до това да се типизират и запомнят съответните двоични дървета. Например за всеки оператор за присвояване се запомня името на променливата и дървото съответстващо на този оператор за присвояване. Допълнително можем да помним и стойността на израза стоящ отдясно в оператора за присвояване.

В последващи цитирания на променливата (в нови изрази или оператори за извеждане) вече може да се изчисли стойността на цитираната променлива (обхождайки по подходящ начин съпоставеното й дърво) или да използваме наготово запомнената стойност.

**ЗАБЕЛЕЖКА: Ако се разграничим от това, че целите числа в нашия език са от тип unsigned long int и въведем изискването за произволно големи (разбира се в разумни граници – например максимум 256 или 512 знака) цели положителни числа.**