You are inside question view of Max Result of Prefix Expression

* **Test ended**

**[Skip to main content](https://www.hackerrank.com/test/7f4e42go3oo/questions/84dfrdobnah" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/7f4e42go3oo/questions)
* [**1**](https://www.hackerrank.com/test/7f4e42go3oo/questions/84dfrdobnah)

**1. Max Result of Prefix Expression**

**Introduction**

For this test you are expected to write a function that evaluates an expression written in prefix notation and returns a value.

**Prefix Notation**

Prefix notation (also known as Polish notation) is an alternative to the more familiar Infix notation.

In infix notation, operators (add, multiply, etc.) are written between their operands (numbers, variables, or sub-expressions).

In prefix notation, operators are written before their operands.

Some examples follow of expressions in infix notation and their equivalents in prefix notation.

In this example the operator is + and its operands are 1 and 2:

Infix expression: 1 + 2

Prefix expression: + 1 2

Value: 3

In this example, the sub-expression + 1 2 is the first operand of the \* operator:

Infix expression: (1 + 2) \* 3

Prefix expression: \* + 1 2 3

Value: 9

Sometimes there are multiple sub-expressions:

Infix expression: ((1 + 2) \* 3) - 4

Prefix expression: - \* + 1 2 3 4

Value: 5

Sub-expressions can be nested arbitrarily deeply:

Infix expression: 6 + ((4 - (2 + 3)) \* 8)

Prefix expression: + 6 \* - 4 + 2 3 8

Value: -2

**Variables**

So far we have only considered numbers as our operands for the prefix expression. For this test we would like your function to also support variables within the expression.

To keep things interesting each variable has a range of possible values. The range does **NOT** include the upper bound.

For example:

expression: + 10 x

variables: { "x": (3, 4) }

value: 13

In this example the only possible value for x is 3, resulting in the expression + 10 3.

In the case where a variable range defines more than one possible value, the function should return the **maximum**value of the expression, for all combinations of variables.

expression: + 10 x

variables: { "x": (3, 7) }

value: 16

In this example x can be 3, 4, 5 or 6, so the expression can be any of

+ 10 3

+ 10 4

+ 10 5

+ 10 6

The last of these has the maximum value, 16.

**Task**

Implement a function **max\_result\_expression(expression, variables)**

that takes as inputs:

* expression, a string containing an expression in prefix notation that might contain variables
* variables, a dictionary containing a range of values for each variable in the expression

and returns:

* the maximum result of the expression for any combination of the given variable values
* None if the expression is invalid or if the expression does not have any valid result.

Our syntax supports 4 operators: +, -, \*, and /. These are the standard arithmetic operators.

* **Note:** / **denotes integer division**, that is / 7 2 evaluates to 3, not 3.5.

The only accepted numeric operands are positive integers in base 10 (e.g. 1, 22, 85 are valid, -1, 0x43, 0, 012 are not valid)

An operator **MUST** have exactly two operands, otherwise the expression is invalid.

A valid variable name is any sequence of characters that does not include whitespace (spaces, tabs, newlines, etc.).

**Examples**

expression: + 1 5

variables: {}

max\_result\_expression: 6​​

expression: + 1 2 3

variables: {}

max\_result\_expression: None

expression: + 1

variables: {}

max\_result\_expression: None

expression: 9

variables: {}

max\_result\_expression: 9

expression: \* + 1 2 3

variables: {}

max\_result\_expression: 9

expression: + 6 \* - 4 + 2 3 8

variables: {}

max\_result\_expression: -2

Operators and operands must be separated by one or more white spaces:

expression: -+1 5 3

variables: {}

max\_result\_expression: None

expression: + 1 2

variables: {}

max\_result\_expression: 3

Expression:

expression: \* + 2 x y

variables: { "x": (0, 2), "y": (2, 4) }

max\_result\_expression: 9

As worked example of the above, for all combinations of each variable's allowed values, the expression has 4 possible results:

| **x** | **y** | **result** |
| --- | --- | --- |
| 0 | 2 | 4 |
| 0 | 3 | 6 |
| 1 | 2 | 6 |
| 1 | 3 | 9 |

From all the possible results, the function must return the maximum value; in this case 9.

Language: Python 3Autocomplete Error (Please refresh page)

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

from typing import Dict, Optional, Tuple

# Complete the max\_result\_expression function below.

# You may add any imports you require from the

standard library.

# Feel free to define your own helper functions,

classes etc as you see fit.

def max\_result\_expression(expression: str, variables:

Dict[str, Tuple[int, int]]) -> Optional[int]:

    """

    Evaluates the prefix expression and calculates the

maximum result for the given variable ranges.

    Arguments:

        expression: the prefix expression to evaluate.

        variables: Keys of this dictionary may appear

as variables in the expression.

            Values are tuples of `(min, max)` that

specify the range of values of the variable.

            The upper bound `max` is NOT included in

the range, so (2, 5) expands to [2, 3, 4].



Line: 4 Col: 1

Test Results

Custom Input

Run Code

Run Tests

Submit