{...} TheBrackets

programing language :)

1) Running programs

```
TheBrackets *.ps
```

2) Print communique

example:

```
?Hello world!?|
```

You should watch:

```
Hello world!
```

We write text between '?' operators, if we want to go to the next line we use ";" or "|" otherwise.

If you want to write '?' as printed character, write '\?', or if you want write '\' (e.g. before the end of the text), write '\\', otherwise character '\' will be printed normally.

example:

```
?What's your name\??;
?That's how we write '\?' character: '\\\?'?;
?\text\\?;
```

You should watch:

```
What's your name?
That's how we write '?' character: '\?'
\text\
```

3) Variables and expressions

We can declare variables two ways:

- Assign

for example: `n:10`

- Read from keyboard

  for example: `$n`

When we assign something to variable we can use a single constant or the whole expression.
If variable earlier existed, it will be update, else it will be create.

Operator priorities in expressions:

1	*subexpression in parentheses*	*()*
2	*multiplication, division, modulo and power operator*	*, /, %, ^
3	*sum and difference operator*	+, -
4	*logic operator*	=, >, <, !

To do print variable value we use '#' character.

example:

```
a: 10
#a ??;
a: a+1
#a ??;
b: a>3
#b ??;
c: a=(10-2.5)
#c ??;
d: b!c
#d ??;
e: a^2
#e ??;
f: (a-3)*10-3+(2.3/2)
#f ??;
g: 8
g: a%8
#g
```

You should watch:

```
10
11
1
0
1
121
78.15
3
```

For logical expressions 1 is true and 0 is false. The '%' operator is interesting, it determines the remainder of the division, but first converts both values into integers. If you know about it you can convert for example, number 1.67 to 1 (cut the value after the decimal point).

for example:

```
value: 1.67
?value = ?| #value ??;
value: value%(value+1)
?value = ?| #value ??;
```

You should watch:

```
value = 1.67
value = 1
```

4) Conditional instructions and loops

- condition
  for example: `@ 1<2 , ?true?; , ?false?; &`
- loop
  for example: `i:3 [ i>0 , { #i ? ?| i:i-1 } ]`

If after any comma you plane put more than one instruction that place these inside '{' and '}' brackets...

You can use additional instructions, '`' who end the loops and '~' who do nothing (for example, when you do not want to do anything in a given place but there must be a instruction there - conditional or loop)

5) Program examples

the parity of numbers:

```
?enter n: ?|
$n
n:n%(n+1)
i:1
n:n+1
[i<n ,
{{ @ (i%2)=0 , { {?number ?|} {#i} {? is parity?;} } , {{ ?number ?|}
{#i} {? in not parity?; }} & } { i:i+1 }}
]
```

prime numbers:

```
?enter n: ?|
$n
n:n%(n+1)
```

```
@ n>0 , {
@ n=1 , { ?1 is not a prime number?; }, { @ n=2 , { ?2 is prime
number?; } , {

i:n-1
prime:1
[ i>1 , {
@ n%i , ~ , { prime:0 ` } &
i : i-1
}]

@ prime , { #n ? is a prime number?; } , { #n ? is not a prime number?;
} &

} & } &

},

{ ?n must be greater than zero! ?; } &
```

fibonacci sequence:

```
?enter n: ?|
$n
n:n%(n+1)

@ n>0 , {
@ n=1 , { ?1 ?| }, { @ n=2, { ?1 1 ?| } , {

?1 1 ?|

a:1
b:1
c:0

i:2
[ i<n ,{

c:a+b
{ #c ? ?| }
a:b
b:c
i:i+1

}]

} & } &
}, { ?n must be greater than zero! ?; } &
```