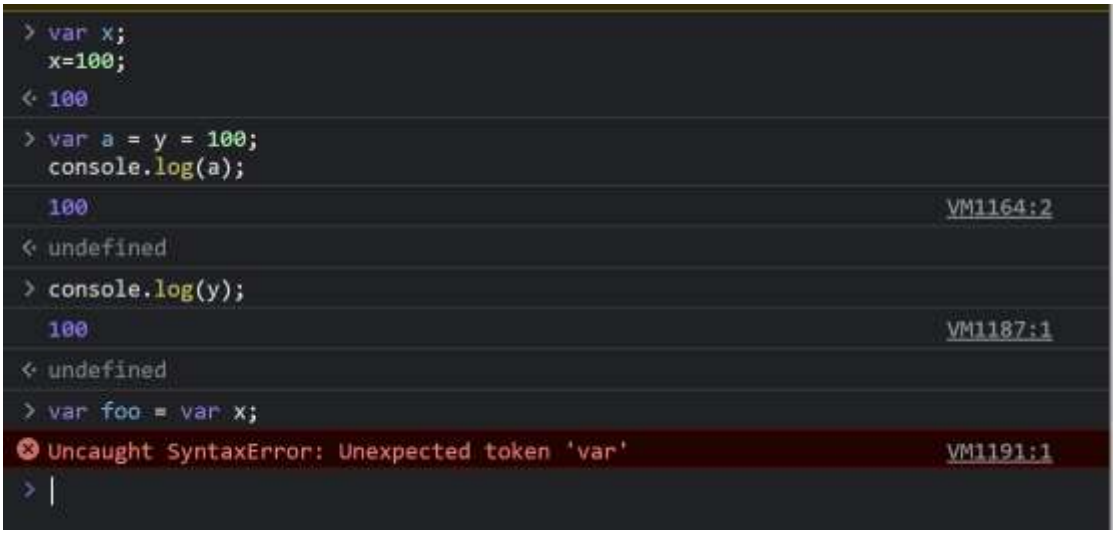
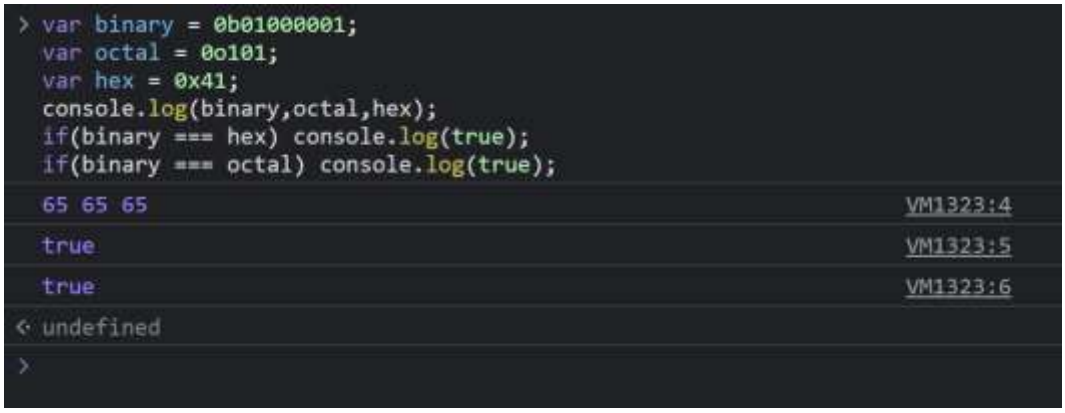


첫번째야 당연히 될것이고 두번째는 c언어나 자바에서는 int a=y=100;이 되진 않는다 하지만
Int a,y;
a=y=100;
은 가능하다 그렇기에 그냥 자바스크립트는 이게 가능 하다라고 외워둠
세번째도 c언어에서 안되는 방법이다
Int a=int b;
당연히 안된다고 생각하기에 안된다고 외워둘뿐 즉 세번째가 표현식이 틀리다!



역시 자바스크립트다 숫자관련은 전부다 number로 보는 듯 하다.

```
> console.log(1 === 1.0);
  console.log(4 / 2);
  console.log(3 / 2)
  true                                                    VM1327:1
  2                                                       VM1327:2
  1.5                                                     VM1327:3
< undefined
>
```

정수와 실수의 데이터타입은 같고 나누기 값이 소수든 정수든 상관없이 잘나옴 심지어 ; (세미콜론)이 없어도 잘나옴

```
> console.log(2**1025);
  Infinity                                                VM1743:1
< undefined
> console.log(2**1024);
  Infinity                                                VM1772:1
< undefined
> console.log(2**1023);
  8.98846567431158e+307                                   VM1803:1
< undefined
>
```

```
> console.log(-(2**1023));
  -8.98846567431158e+307                                  VM2568:1
< undefined
> console.log(-(2**1024));
  -Infinity                                               VM2573:1
< undefined
>
> console.log(+'a');
  NaN                                                     VM3002:1
< undefined
> console.log('a'+);
✖ Uncaught SyntaxError: Unexpected token ')'              VM3010:1
>
```

무한대는 2의 1024승 정도면 infinity로 인식하는듯 추가적인 계산결과 한번 infinity가 되면 계속 infinity인듯

```
> var NAN=1;
< undefined
> console.log(NAN);
  1                                                    VM3128:1
< undefined
> var nan=2;
< undefined
> console.log(nan);
  2                                                    VM3251:1
< undefined
>
```

왜인진 모르겠는데 잘나온다

```
> console.log(a);
  여거랑'여거랑'                                        VM3587:1
< undefined
> console.log(b);
  꽃게랑"꽃게랑"                                        VM3646:1
< undefined
>
```

일단 전부 문자열로 본다 애초에
‘문자’가 없는듯
‘문자열 ' 이 있는듯
그리곤 ‘’ “”잘나온다
둘다 문자열로 보나?

```
> console.log(typeof("여거랑'여거랑'"));
  string                                               VM3435:1
< undefined
> console.log(typeof('여거랑"여거랑"'));
  string                                               VM3448:1
< undefined
> console.log(typeof("123"));
  string                                               VM3460:1
< undefined
> console.log(typeof('123'));
  string                                               VM3488:1
< undefined
> console.log(typeof(`1`));
  string                                               VM3502:1
< undefined
> console.log(typeof("1"));
  string                                               VM3525:1
< undefined
```

```
> console.log(`a + b = ${1 + 2}`);
  console.log('a + b = ${1 + 2}');
  a + b = 3                                            VM3657:1
  a + b = ${1 + 2}                                     VM3657:2
< undefined
>
```

‘123’은 안에있는 모든걸 문자열로 인식하는듯함 하지만 `123`은 중간에 ${}가있다면 c언어의 “%d”같은 느낌이랄까?

의도적 부재를 왜 사용할까? 몰?루

과연 아래의 사용법이 옳은 선택일까? 다른 방법으로 변수를 소멸시키는게 좋지 않을까?

```
var night = 'Turtle';
// 밑의 선언으로 인해 night는 더이상 터틀이라는 값을 참조하지 않으며 언젠가 gc에 없어져버린다.
night = null;
```

NULL이 맞다고 생각한다
일단 기본적으로 글자수가 짧다
그렇기에 실수할 확률도 적을뿐더러
전체적인 코드 bite가 작아서 뭔가
기분이 좋다

ECMAScript 사양은 문자열과 숫자 타입 외에는 명시적으로 규정하고 있지 않은데 그렇다면 해당 데이터 타입들 외에는 어떤 식으로 계산되고 있는가?

```
> console.log(1 + NaN);
  NaN
< undefined
> console.log(NaN == 1);
  false
< undefined
> console.log(NaN < 1);
  false
< undefined
> console.log(NaN > 1);
  false
< undefined
```

```
> var x;
< undefined
> console.log(x);
  undefined
< undefined
> console.log(undefined > 1);
  false
< undefined
> console.log(1 + undefined);
  NaN
< undefined
> console.log(undefined == 1);
  false
< undefined
```

💡 심벌 테이블 이라는 뜻을 알아보시오

컴파일러든 인터프리터든 변수나
함수같은것을 저장하는 데이터 구조다
이것이 컴파일러다 인터프리터를
실행할때 효율적으로 실행시켜주는듯함

💡 대표적인 동적/정적 언어를 조사해보시오

동적언어:C,C++,Java,Rust
정적언어:파이썬,JavaScript

```
> var a = '1';
  console.log(+a, typeof +a);
  console.log(a, typeof a);
  a = true;
  console.log(+a, typeof +a);
  console.log(a, typeof a);
  a = false;
  console.log(+a, typeof +a);
  console.log(a, typeof a);
  a = 'Hi';
  console.log(+a, typeof +a);
  console.log(a, typeof a);
  1 'number'                    VM3754:2
  1 string                      VM3754:3
  1 'number'                    VM3754:5
  true 'boolean'                VM3754:6
  0 'number'                    VM3754:8
  false 'boolean'               VM3754:9
  NaN 'number'                  VM3754:11
  Hi string                     VM3754:12
< undefined
```

+a는 숫자고 a는 문자열인게 신기하네
‘1’이렇게 문자열로 지정해도 +가붙으면
넘버로 바뀌고 안붙으면 계속 문자열인가 봄

암묵적 타입 변환 또는 **타입 강제 변환**에 대해서 알아보시오

+연산을 할때
문자열이 있으면
무조건 문자열로
인식하는듯
다른 */는 문자열이
있어도 숫자로
인식함
(암묵적 타입변환)

```
console.log("10"+1);
console.log("10"*10);
```

```
101
100
```

```
let a = 1;
let b = "2";
undefined
console.log(a + b);
12
undefined
console.log(a + Number(b));
3
undefined
```

Number(변수명)을하
면 타입을 강제로
변환할수 있는듯
하다
(타입 강제 변환)

5=='5'가 true인건
좀특이하다 암묵적
타입 변환인가?

쨋든 5==='5'가
false인건 당연히
타입이 다르기에

```
> 5 == 5;
< true
> 5 == '5';
< true
> 5 === 5;
< true
> 5 === '5';
< false
```

```
> '0' == '';
< false
> 0 == '';
< true
> 0 == '0';
< true
> false == 'false';
< false
> false == '0';
< true
```

'0'==''이 틀린건 당연하다고 생각하고 있는데0==''이 true인걸 보고 놀랐다 이게 왜맞지?
0==''=='0'이 되어야 하는게 아닌가? 이해할수 없다

'false'가정의하고 있는건 0인걸 알기에 납득이 된다

```
> false == null;
< false
> false == undefined;
< false
> NaN === NaN
< false
> 0 == -0
< true
> 0 === -0
< true
```

0이나 -0이나 당연히 같겟지 생각은 하는데
NaN===NaN은 왜false지?

```
> - 0 === 0 ;
< true
> Object.is(-0,0)
< false
> NaN === NaN;
< false
> Object.is(NaN,NaN);
< true
> Object.is(0,-1*0)
< false
> Object.is(-0,-1*0)
< true
```

-0==0도 true긴한데
저 object는 -0이나 0을 false로 보나보다 왜지?
NAN은 왜 같다고 뜨는데 왜 타입은 다른데 NAN이라 그런가

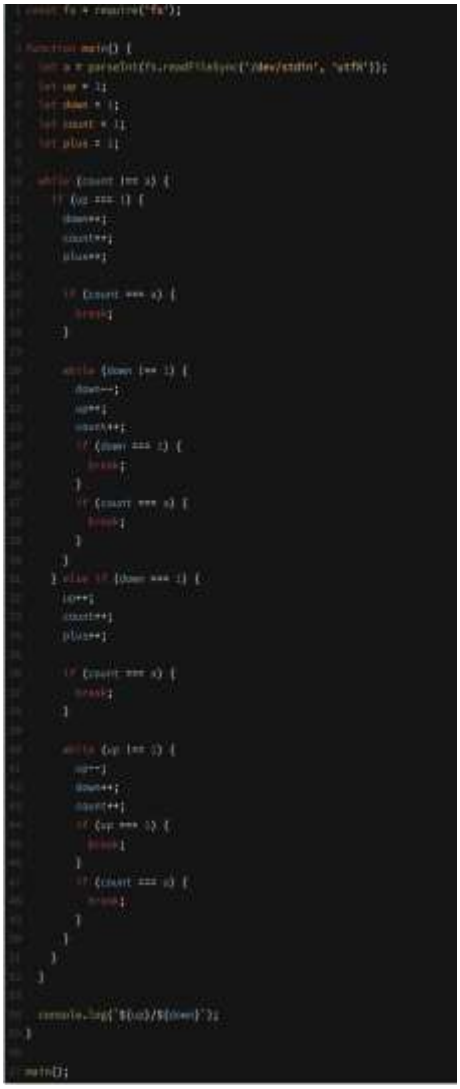
```
> console.log(typeof("Hello, World!"));
  string
< undefined
> console.log(typeof(123));
  number
< undefined
> console.log(typeof(true));
  boolean
< undefined
> console.log(typeof(undefined));
  undefined
< undefined
> console.log(typeof(Symbol("avc")));
  symbol
< undefined
> console.log(typeof({}));
  object
< undefined
```

```
> function abc{};
✕ Uncaught SyntaxError: Unexp
> function abc(value){
      console.log(1);
  }
< undefined
> console.log(typeof(abc));
  function
```

```
1 console.log("Hello World!");
```

66902512	rhkdqns68	2557	맞았습니다!!	9272 KB	108 ms	node.js / 수정	28 B	10초 전
66902048	rhkdqns68	1193	맞았습니다!!	9588 KB	188 ms	node.js / 수정	852 B	11분 전

간단한건 잘되는데
조금만 복잡해져도
틀렸다가 뜬다 왜지
AI한테 맡겨서
바꿔보면 또 맡다고
뜨는데 백준에
올려서 맞출려면
뭔갈 더
써넣어야하나?



이 밑에는 infinity가
어디서부터 터지는지
실험한거

```
#include<stdio.h>
 int n;
 int main()
 {
 scanf("%d", &n);
 printf("console.log(21023");
for (int i = 1; i <= 53; i++)
 {
       printf("+2%d",1023 - i);
 }
//for (int i = 54; i <= n; i++)
 //{
 //    printf("+2**%d", 1023 - i);
//}
printf(")");
 return 0;
 }
```

53부터 터지는걸 알수있음

9.979201547673599 " 0 " 582818635651842e+291

이숫자부터 저 0부분을 줄이려고하면 안줄어듬