

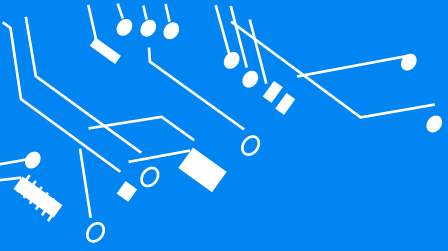


C programming

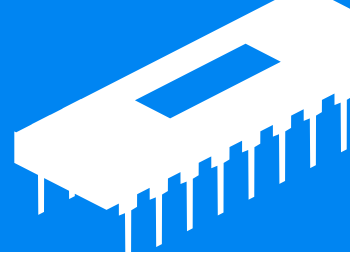
Exercises

A glowing blue microchip is centered on a circuit board. The chip has a bright blue, textured surface and is surrounded by glowing blue lines and dots that resemble circuit traces and data points. The background is dark blue with a subtle pattern of glowing dots and lines.

Problem Solving



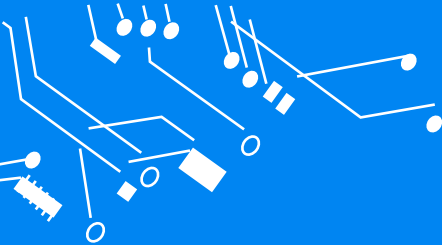
Q1



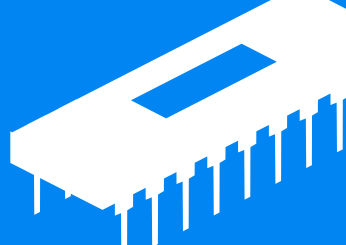
Write a C program to swap 2 variables without using a third temporary variable.

Function prototype :

```
void swapVariables (int* first, int* second);
```



Q2



Write a C function that counts the number of 1's in an unsigned 32-bit number.

Example :

0x89F00123

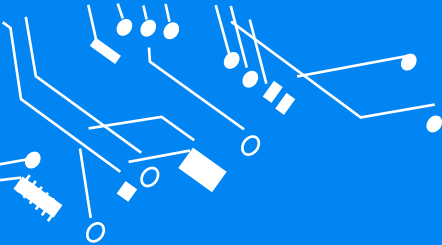
will be :

1000 1001 1111 0000 0000 0001 0010 0011

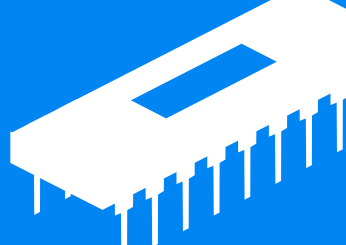
=> 11 ones

Function prototype :

```
int countOnes(unsigned int n);
```



Q3



Louise joined a social networking site to stay in touch with her friends. The signup page required her to input a *name* and a *password*. However, the password must be *strong*. The website considers a password to be *strong* if it satisfies the following criteria:

- Its length is at least 6 digits.
- It contains at least one digit.
- It contains at least one lowercase English character.
- It contains at least one uppercase English character.
- It contains at least one special character. The special characters are:

!@#\$%^&*()-+

She typed a random string of length in the password field but wasn't sure if it was strong. Given the string she typed, can you find the minimum number of characters she must add to make her password strong?

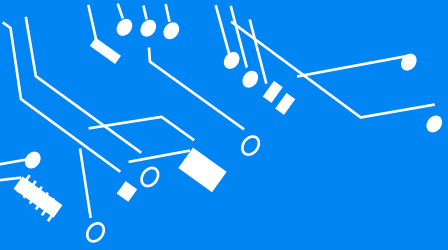
Note: Here's the set of types of characters in a form you can paste in your solution:

```
numbers = "0123456789"
```

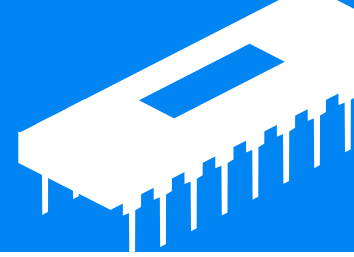
```
lower_case = "abcdefghijklmnopqrstuvwxyz"
```

```
upper_case = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
special_characters = "!@#$%^&*()-+"
```



Q3 continues



Input to your function contain the password length n , and the password .

Constraints :

$1 \leq n \leq 100$

Example :

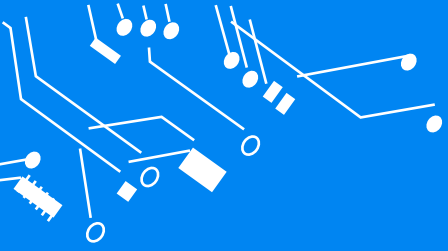
3

Ab1

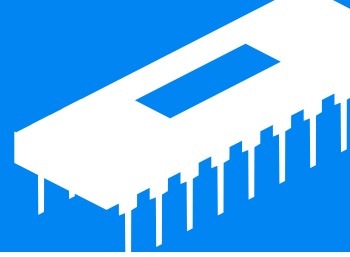
output = 3

Function prototype :

```
int minimumNumber(int n, char* password);
```



Q4



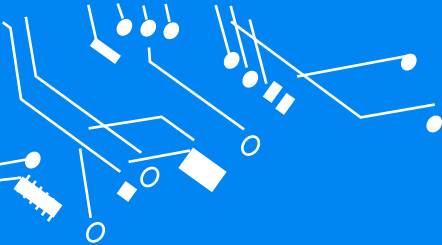
Given a sentence, print each word of the sentence in a new line.

Sample Input 0

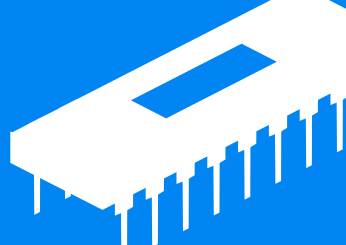
```
This is C
```

Sample Output 0

```
This  
is  
C
```

Q5



Given an integer n , return *true* if it is a power of two. Otherwise, return *false*.

An integer n is a power of two, if there exists an integer x such that $n == 2^x$.

Note : ($-2^{31} \leq n \leq 2^{31} - 1$)

`bool isPowerOfTwo(int n);`

Example 1:

Input: $n = 1$
Output: true
Explanation: $2^0 = 1$

Example 3:

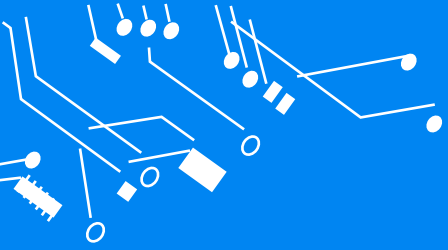
Input: $n = 3$
Output: false

Example 2:

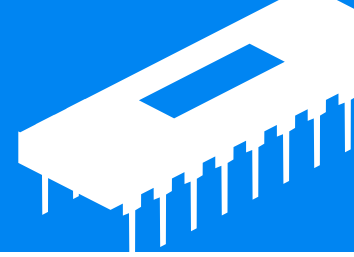
Input: $n = 16$
Output: true
Explanation: $2^4 = 16$

Example 4:

Input: $n = 4$
Output: true



Q6



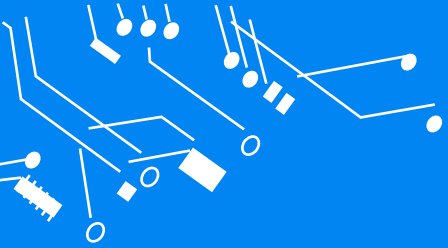
There is a function that returns:

- 1 if x is positive.
- -1 if x is negative.
- 0 if x is equal to 0.

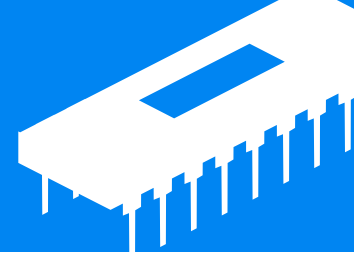
You are given an integer array `nums`. Let product be the product of all values in the array `nums`.

Find the sign of the product .

```
int arraySign(int* nums, int numsSize);
```



Q6 continues



Example 1:

Input: `nums = [-1,-2,-3,-4,3,2,1]`

Output: 1

Example 2:

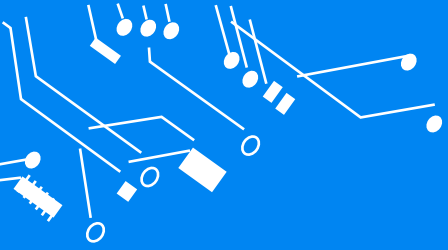
Input: `nums = [1,5,0,2,-3]`

Output: 0

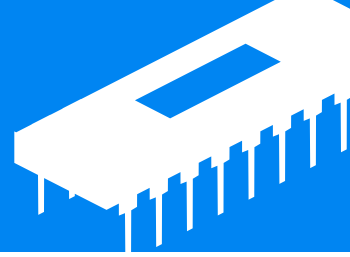
Example 3:

Input: `nums = [-1,1,-1,1,-1]`

Output: -1



Q7



Write a C program to find the longest occurrence of an even word (number of characters is even)in a given sentence.

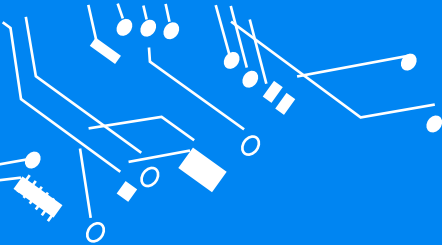
Example :

input : “welcome to Computer world, Programming is Fun”

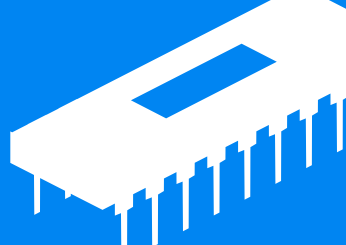
output : “Computer”

Function prototype :

`char* LongestEvenWord(char* sentence)`



Q8



Given two non-negative integers low and high.
Return the *count of odd numbers between low and high (inclusive)*.

$(0 \leq \text{low} \leq \text{high} \leq 10^9)$

```
int countOdds(int low, int high);
```

Example 1:

Input: low = 3, high = 7

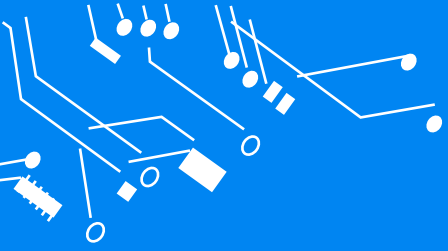
Output: 3

Explanation: The odd numbers between 3 and 7 are [3,5,7].

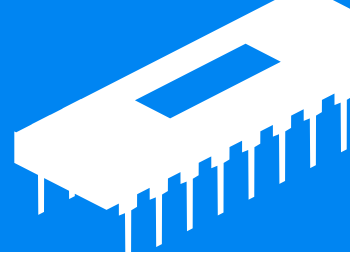
Example 2:

Input: low = 8, high = 10

Output: 1

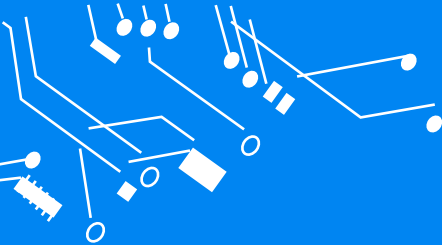


Q9

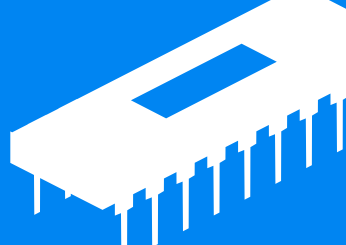


Write a C function to convert a hexadecimal number to decimal

```
int hexToDecimal(char * hex);
```



Q10

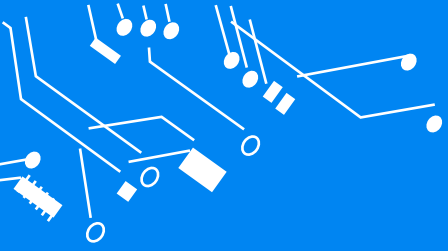


Write an efficient algorithm that searches for a value in an $m \times n$ matrix. This matrix has the following properties:

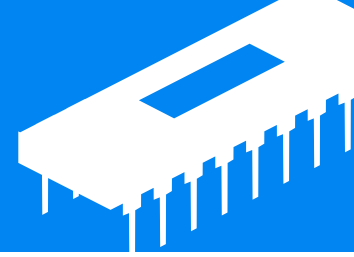
Integers in each row are sorted from left to right.

The first integer of each row is greater than the last integer of the previous row.

```
bool searchMatrix(int** matrix, int matrixSize, int*  
matrixColSize, int target);
```



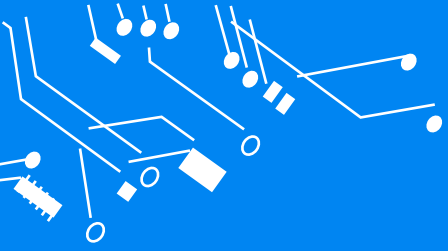
Q10 continue



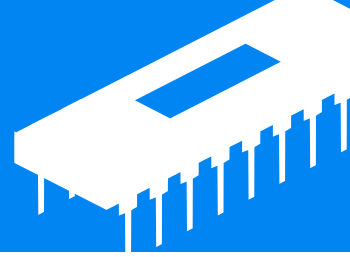
| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 10 | 11 | 16 | 20 |
| 23 | 30 | 34 | 60 |

Input: matrix = [[1,3,5,7],[10,11,16,20],
[23,30,34,60]], target = 3

Output: true

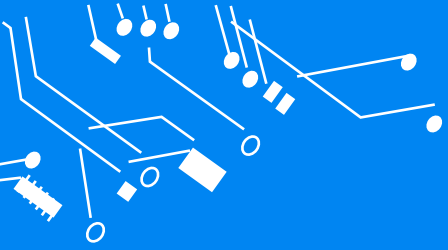


Q11

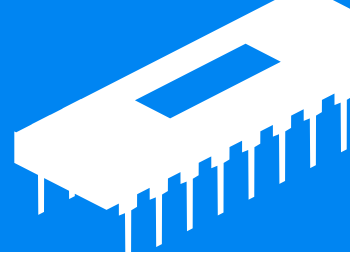


Write a C function to reverse a sentence

```
char* reverseSentence(char * str);
```

Q12



Write a C function that returns the count of the longest consecutive occurrence of a given number in an array .

Example :

array = {1,2,2,3,3,3,3,4,4,4,3,3,1} →
searching for 3 , output = 4

Function prototype :

```
int FindLongest(int size, int * arr, int number);
```