

CONDITIONAL STATEMENTS IN C :



HackerRank

Prepare

C

Conditionals and Loops

Conditional Statements in C

Problem

Task

Given a positive integer denoting n , do the following:

- If $1 \leq n \leq 9$, print the lowercase English word corresponding to the number (e.g., one for 1, two for 2, etc.).
- If $n > 9$, print Greater than 9.

Input Format

The first line contains a single integer, n .

Constraints

- $1 \leq n \leq 10^9$

Output Format

If $1 \leq n \leq 9$, then print the lowercase English word corresponding to the number (e.g., one for 1, two for 2, etc.); otherwise, print Greater than 9 instead.

Sample Input

5

Sample Output

five

Sample Input #01

Submissions

Leaderboard

Discussions

Sample Input #01

```
8
```

Sample Output #01

```
eight
```

Sample Input #02

```
44
```

Sample Output #02

```
Greater than 9
```

```

10  int main()
11  {
12      int number;
13      scanf("%d",&number);
14      if(number==1){
15          printf("one");
16      }
17      else if(number==2){
18          printf("two");
19      }
20      else if(number==3){
21          printf("three");
22      }
23      else if(number==4){
24          printf("four");
25      }
26      else if(number==5){
27          printf("five");
28      }
29      else if(number==6){
30          printf("six");
31      }
32      else if(number==7){
33          printf("seven");
34      }
35      else if(number==8){
36          printf("eight");
37      }
38      else if(number==9){
39          printf("nine");
40      }
41      else{
42          printf("Greater than 9");
43      }
44      return 0;
45  }
46

```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✔ Sample Test case 0

Input (stdin)

[Download](#)

✔ Sample Test case 1

1 5

✔ Sample Test case 2

Your Output (stdout)

1 five

Expected Output

[Download](#)

1 five

For Loop in C ★

Task

For each integer n in the interval $[a, b]$ (given as input) :

- If $1 \leq n \leq 9$, then print the English representation of it in lowercase. That is "one" for **1**, "two" for **2**, and so on.
- Else if $n > 9$ and it is an even number, then print "even".
- Else if $n > 9$ and it is an odd number, then print "odd".

Input Format

The first line contains an integer, a .

The second line contains an integer, b .

Constraints

$$1 \leq a \leq b \leq 10^6$$

Output Format

Print the appropriate English representation, `even`, or `odd`, based on the conditions described in the 'task' section.

Note: $[a, b] = \{x \in \mathbb{Z} \mid a \leq x \leq b\} = \{a, a + 1, \dots, b\}$

Sample Input

```
8
11
```

Sample Output

```
eight
nine
```

```
even
odd
```

```

1  #include <stdio.h>
2  int main()
3  {
4      int a, b, i;
5      scanf("%d\n%d", &a, &b);
6      for(i=a; i<=b; i++){
7          if (i<10) {
8              switch(i) {
9                  case 1: printf("one\n"); break;
10                 case 2: printf("two\n"); break;
11                 case 3: printf("three\n"); break;
12                 case 4: printf("four\n"); break;
13                 case 5: printf("five\n"); break;
14                 case 6: printf("six\n"); break;
15                 case 7: printf("seven\n"); break;
16                 case 8: printf("eight\n"); break;
17                 case 9: printf("nine\n"); break;
18             }
19         }
20         else if(i>9 && i%2==0)
21             printf("even\n");
22         else if(i>9 && i%2!=0)
23             printf("odd\n");
24     }
25     return 0;
26 }

```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

Input (stdin)

[Download](#)

```

1  8
2  11

```

Your Output (stdout)

```

1  eight
2  nine
3  even
4  odd

```

Expected Output

[Download](#)

```

1  eight

```



Problem

Print a pattern of numbers from 1 to n as shown below. Each of the numbers is separated by a single space.

```
4 4 4 4 4 4 4
4 3 3 3 3 3 4
4 3 2 2 2 3 4
4 3 2 1 2 3 4
4 3 2 2 2 3 4
4 3 3 3 3 3 4
4 4 4 4 4 4 4
```

Submissions

Input Format

The input will contain a single integer n .

Constraints

$$1 \leq n \leq 1000$$

Sample Input 0

```
2
```

Leaderboard

Sample Output 0

```
2 2 2
2 1 2
2 2 2
```

Discussions

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <math.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8
9      int n,i,j;
10     scanf("%d", &n);
11
12     for(i=1;i<=2*n-1;i++){
13         for(j=1;j<=2*n-1;j++){
14             int min_val = i < j ? i : j;
15             min_val = min_val < (2 * n - i) ? min_val : (2 * n - i);
16             min_val = min_val < (2 * n - j) ? min_val : (2 * n - j);
17
18             printf("%d ", n - min_val + 1);
19         }
20         printf("\n");
21     }
22     return 0;
23 }
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

Input (stdin)

[Download](#)

```
1 2
```

✓ Sample Test case 1

✓ Sample Test case 2

Your Output (stdout)

```
1 2 2 2
2 2 1 2
3 2 2 2
```

Expected Output

[Download](#)

```
1 2 2 2
2 2 1 2
3 2 2 2
```

H HackerRank | **Prepare** > **C** > **Conditionals and Loops** > **Sum of Digits of a Five Digit**

Problem

Objective

The modulo operator, `%`, returns the remainder of a division. For example, `4 % 3 = 1` and `12 % 10 = 2`. The ordinary division operator, `/`, returns a truncated integer value when performed on integers. For example, `5 / 3 = 1`. To get the last digit of a number in base 10, use `10` as the modulo divisor.

Task

Given a five digit integer, print the sum of its digits.

Input Format

The input contains a single five digit number, n .

Constraints

$10000 \leq n \leq 99999$

Output Format

Print the sum of the digits of the five digit number.

Sample Input 0

```
10564
```

Sample Output 0

```
16
```

Submissions

Leaderboard

Discussions

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <math.h>
4  #include <stdlib.h>
5  int main() {
6      int n,i,sum=0;
7      scanf("%d", &n);
8  if (n < 10000 || n > 99999){
9      printf("Invalid number");
10 }
11 while(n>0){
12     sum+=n%10;
13     n/=10;
14 }
15     printf("%d",sum);
16     return 0;
17 }

```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

Input (stdin)

[Download](#)

1	10564
---	-------

Your Output (stdout)

1	16
---	----

Expected Output

[Download](#)

1	16
---	----



In this challenge, you will use logical bitwise operators. All data is stored in its binary representation. The logical operators, and C language, use **1** to represent true and **0** to represent false. The logical operators compare bits in two numbers and return true or false, **0** or **1**, for each bit compared.

- **Bitwise AND operator &** The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0. It is denoted by &.
- **Bitwise OR operator |** The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. It is denoted by |.
- **Bitwise XOR (exclusive OR) operator ^** The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

For example, for integers 3 and 5,

```
3 = 00000011 (In Binary)
5 = 00000101 (In Binary)

AND operation      OR operation      XOR operation
00000011          00000011          00000011
& 00000101        | 00000101        ^ 00000101
-----          -----          -----
00000001 = 1      00000111 = 7      00000110 = 6
```

You will be given an integer n , and a threshold, k . *For each number i from 1 through n , find the maximum value of the logical and, or and xor when compared against all integers j through n that are greater than i . Consider a value only if the comparison returns a result less than k .* Print the results of the and, or and exclusive or comparisons on separate lines, in that order.



Example

$n = 3$

$k = 3$

The results of the comparisons are below:

```
a b  and or xor
1 2   0  3  3
1 3   1  3  2
2 3   2  3  1
```

For the **and** comparison, the maximum is **2**. For the **or** comparison, none of the values is less than k , so the maximum is **0**. For the **xor** comparison, the maximum value less than k is **2**. The function should print:

```
2
0
2
```

Function Description

Complete the `calculate_the_maximum` function in the editor below.

`calculate_the_maximum` has the following parameters:

- `int n`: the highest number to consider
- `int k`: the result of a comparison must be lower than this number to be considered

Prints

Print the maximum values for the and, or and xor comparisons, each on a separate line.

Input Format

The only line contains 2 space-separated integers, n and k .

Constraints

- $2 \leq n \leq 10^3$
- $2 \leq k \leq n$

Sample Input 0

```
5 4
```

Sample Output 0

```
2
3
3
```

Explanation 0

$n = 5, k = 4$

$S = \{1, 2, 3, 4, 5\}$

All possible values of a and b are:

1. $a = 1, b = 2$; $a \& b = 0$; $a | b = 3$; $a \oplus b = 3$;

2. $a = 1, b = 3$; $a \& b = 1$; $a | b = 3$; $a \oplus b = 2$;

3. $a = 1, b = 4$; $a \& b = 0$; $a | b = 5$; $a \oplus b = 5$;

4. $a = 1, b = 5$; $a \& b = 1$; $a | b = 5$; $a \oplus b = 4$;

5. $a = 2, b = 3$; $a \& b = 2$; $a | b = 3$; $a \oplus b = 1$;

6. $a = 2, b = 4$; $a \& b = 0$; $a | b = 6$; $a \oplus b = 6$;

7. $a = 2, b = 5$; $a \& b = 0$; $a | b = 7$; $a \oplus b = 7$;

8. $a = 3, b = 4$; $a \& b = 0$; $a | b = 7$; $a \oplus b = 7$;

9. $a = 3, b = 5$; $a \& b = 1$; $a | b = 7$; $a \oplus b = 6$;

10. $a = 4, b = 5$; $a \& b = 4$; $a | b = 5$; $a \oplus b = 1$;

- The maximum possible value of $a \& b$ that is also $< (k = 4)$ is 2, so we print 2 on first line.
- The maximum possible value of $a | b$ that is also $< (k = 4)$ is 3, so we print 3 on second line.
- The maximum possible value of $a \oplus b$ that is also $< (k = 4)$ is 3, so we print 3 on third line.

```

#include <stdio.h>
void calculate_the_maximum(int n, int k) {
    int max_and = 0, max_or = 0, max_xor = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            int current_and = i & j;
            int current_or = i | j;
            int current_xor = i ^ j;
            if (current_and > max_and && current_and < k) {
                max_and = current_and;
            }
            if (current_or > max_or && current_or < k) {
                max_or = current_or;
            }
            if (current_xor > max_xor && current_xor < k) {
                max_xor = current_xor;
            }
        }
    }
    printf("%d\n%d\n%d\n", max_and, max_or, max_xor);
}

int main() {
    int n, k;
    scanf("%d %d", &n, &k);
    calculate_the_maximum(n, k);

    return 0;
}

```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test

✔ Sample Test case 0

Input (stdin)

1	5 4
---	-----

Your Output (stdout)

1	2
2	3
3	3

Expected Output

1	2
2	3
3	3