

Started on Monday, 25 August 2025, 9:50 PM

State Finished

Completed on Monday, 25 August 2025, 10:35 PM

Time taken 45 mins 15 secs

Marks 10.00/10.00

Grade **100.00** out of 100.00

Question 1 | Correct Mark 1.00 out of 1.00

Given a positive integer `n`, write a function that returns the number of set bits in its binary representation (also known as the [Hamming weight](#)).

Example 1:**Input:** `n = 11`**Output:** 3**Explanation:**

The input binary string **1011** has a total of three set bits.

Example 2:**Input:** `n = 128`**Output:** 1**Explanation:**

The input binary string **10000000** has a total of one set bit.

Example 3:**Input:** `n = 2147483645`**Output:** 30**Explanation:**

The input binary string **1111111111111111111111111101** has a total of thirty set bits.

For example:

Input	Result
11	3
128	1

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 c=0;
3 for i in bin(n)[2:]:
4     if i=='1':
5         c+=1
6 print(c)

```

Input	Expected	Got	
✓ 11	3	3	✓
✓ 128	1	1	✓
✓ 32	1	1	✓
✓ 2147483645	30	30	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 2 | Correct Mark 1.00 out of 1.00

A **happy number** is a number defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits.
- Repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1.
- Those numbers for which this process **ends in 1** are happy.

Print `true` if `n` is a happy number, and `false` if not.

For example:

Input	Result
19	True
2	False

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 while n!=1 and n!=4:
3     temp=n
4     sum=0
5     while temp>0:
6         digit=temp%10
7         sum=sum+digit*digit
8         temp=temp//10
9     n=sum
10 if n==1:
11     print(True)
12 else:
13     print(False)

```

	Input	Expected	Got	
✓	19	True	True	✓
✓	2	False	False	✓
✓	82	True	True	✓
✓	16	False	False	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 3 | Correct Mark 1.00 out of 1.00

An **ugly number** is a *positive* integer which does not have a prime factor other than 2, 3, and 5.

Given an integer **n**, Print **True** if **n** is an **ugly number**, Otherwise Print **False**.

For example:

Input	Result
6	True
14	False

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 if n<=0:
3     print(False)
4 else:
5     while n%2==0:
6         n=n//2
7     while n%3==0:
8         n=n//3
9     while n%5==0:
10        n=n//5
11 if n==1:
12     print(True)
13 else:
14     print(False)

```

	Input	Expected	Got	
✓	6	True	True	✓
✓	14	False	False	✓
✓	125	True	True	✓
✓	21	False	False	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 4 | Correct Mark 1.00 out of 1.00

You are climbing a staircase. It takes **n** steps to reach the top.

Each time you can either climb **1** or **2** steps. In how many distinct ways can you climb to the top?

Example 1:

Input: n = 2

Output: 2

Explanation: There are two ways to climb to the top.

1. 1 step + 1 step
2. 2 steps

Example 2:

Input: n = 3

Output: 3

Explanation: There are three ways to climb to the top.

1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

Constraints:

- **1 <= n <= 45**

For example:

Input	Result
2	2
3	3

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 if n==1:
3     print(1)
4 else:
5     a=1
6     b=2
7     i=3
8 while(i<=n):
9     c=a+b
10    a=b
11    b=c
12    i=i+1
13    print(b)

```

	Input	Expected	Got	
✓	2	2	2	✓
✓	3	3	3	✓
✓	4	5	5	✓
✓	5	8	8	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 5 | Correct Mark 1.00 out of 1.00

Given an integer `num`, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

Input: num = 38

Output: 2

Explanation: The process is

38 --> 3 + 8 --> 11

11 --> 1 + 1 --> 2

Since 2 has only one digit, return it.

Example 2:

Input: num = 0

Output: 0

For example:

Input	Result
38	2
0	0

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 while n>=10:
3     temp=n
4     sum=0
5     while temp>0:
6         digit=temp%10
7         sum=sum+digit
8         temp=temp//10
9     n=sum
10 print(n)

```

	Input	Expected	Got	
✓	38	2	2	✓
✓	0	0	0	✓
✓	11	2	2	✓
✓	50	5	5	✓
✓	81	9	9	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 6 | Correct Mark 1.00 out of 1.00

Given an integer n , return *the number of trailing zeroes in $n!$* .

Note that $n! = n * (n - 1) * (n - 2) * \dots * 3 * 2 * 1$.

Example 1:

Input: $n = 3$

Output: 0

Explanation: $3! = 6$, no trailing zero.

Example 2:

Input: $n = 5$

Output: 1

Explanation: $5! = 120$, one trailing zero.

Example 3:

Input: $n = 0$

Output: 0

Constraints:

- $0 \leq n \leq 10^4$

For example:

Input	Result
3	0
5	1

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 c=0
3 while(n>0):
4     n=n//5
5     c+=n
6 print(c)

```

	Input	Expected	Got	
✓	3	0	0	✓

	Input	Expected	Got	
✓	5	1	1	✓
✓	0	0	0	✓
✓	10	2	2	✓
✓	25	6	6	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 7 | Correct Mark 1.00 out of 1.00

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} element, return 0.

Example

$n = 20$

$p = 3$

The factors of 20 in ascending order are {1, 2, 4, 5, 10, 20}. Using 1-based indexing, if $p = 3$, then 4 is returned. If $p > 6$, 0 would be returned.

Constraints

$1 \leq n \leq 10^{15}$

$1 \leq p \leq 10^9$

The first line contains an integer n , the number to factor.

The second line contains an integer p , the 1-based index of the factor to return.

Sample Case 0**Sample Input 0**

10

3

Sample Output 0

5

Explanation 0

Factoring $n = 10$ results in {1, 2, 5, 10}. Return the $p = 3^{\text{rd}}$ factor, 5, as the answer.

Sample Case 1**Sample Input 1**

10

5

Sample Output 1

0

Explanation 1

Factoring $n = 10$ results in {1, 2, 5, 10}. There are only 4 factors and $p = 5$, therefore 0 is returned as the answer.

Sample Case 2**Sample Input 2**

1

1

Sample Output 2

1

Explanation 2

Factoring $n = 1$ results in {1}. The $p = 1^{\text{st}}$ factor of 1 is returned as the answer.

For example:

Input	Result
10	5
3	
10	0
5	

Input	Result
1	1
1	

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 p=int(input())
3 i=1
4 c=0
5 ans=0
6 while(i<=n):
7     if n%i==0:
8         c=c+1
9     if c==p:
10        ans=i
11        break
12    i=i+1
13 if ans!=0:
14     print(ans)
15 else:
16     print(0)

```

	Input	Expected	Got	
✓	10 3	5	5	✓
✓	10 5	0	0	✓
✓	1 1	1	1	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 8 | Correct Mark 1.00 out of 1.00

A **perfect number** is a **positive integer** that is equal to the sum of its **positive divisors**, excluding the number itself. A **divisor** of an integer x is an integer that can divide x evenly.

Given an integer n , return `True` if n is a perfect number, otherwise return `False`.

Example 1:

```
Input: num = 28
Output: True
Explanation: 28 = 1 + 2 + 4 + 7 + 14
1, 2, 4, 7, and 14 are all divisors of 28.
```

Example 2:

```
Input: num = 7
Output: False
```

Constraints:

- `1 <= num <= 108`

Answer: (penalty regime: 0 %)

```
1 n=int(input())
2 if n<=0:
3     print(False)
4 else:
5     tot=0
6     i=1
7 while(i<n):
8     if n%i==0:
9         tot=tot+i
10    i=i+1
11 if tot==n:
12     print(True)
13 else:
14     print(False)
```

	Input	Expected	Got	
✓	28	True	True	✓
✓	7	False	False	✓
✓	8128	True	True	✓
✓	496	True	True	✓
✓	500	False	False	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 9 | Correct Mark 1.00 out of 1.00

Write a program in Python to display a pyramid with "*" as follows,

For example:

Input	Result
4	<pre> * *** **** ***** </pre>

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 for i in range (1,n+1):
3     for _ in range(n-i):
4         print(end=' ')
5     for j in range(2*i-1):
6         print("*",end='')
7     print()

```

	Input	Expected	Got	
✓	4	<pre> * *** **** ***** </pre>	<pre> * *** **** ***** </pre>	✓
✓	2	<pre> * *** </pre>	<pre> * *** </pre>	✓
✓	5	<pre> * *** **** ***** ***** </pre>	<pre> * *** **** ***** ***** </pre>	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 10 | Correct Mark 1.00 out of 1.00

Given an integer `num`, return *the number of digits in num that divide num*.

An integer `val` divides `num` if `num % val == 0`.

Example 1:

Input: `num = 7`

Output: 1

Explanation: 7 divides itself, hence the answer is 1.

Example 2:

Input: `num = 121`

Output: 2

Explanation: 121 is divisible by 1, but not 2. Since 1 occurs twice as a digit, we return 2.

Example 3:

Input: `num = 1248`

Output: 4

Explanation: 1248 is divisible by all of its digits, hence the answer is 4.

For example:

Input	Result
7	1
121	2
1248	4

Answer: (penalty regime: 0 %)

```

1 n=int(input())
2 num=n
3 c=0
4 while n>0:
5     digit=n%10
6     if digit!=0:
7         if num%digit==0:
8             c=c+1
9     n=n//10
10 print(c)

```

	Input	Expected	Got	
✓	7	1	1	✓
✓	121	2	2	✓

	Input	Expected	Got	
✓	1248	4	4	✓
✓	12	2	2	✓
✓	45	1	1	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.