# Problem 343: Integer Break

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

Given an integer

n

, break it into the sum of

k

positive integers

, where

k >= 2

, and maximize the product of those integers.

Return

the maximum product you can get

.

Example 1:

Input:

n = 2

Output:

1

Explanation:

2 = 1 + 1, 1 × 1 = 1.

Example 2:

Input:

n = 10

Output:

36

Explanation:

10 = 3 + 3 + 4, 3 × 3 × 4 = 36.

Constraints:

2 <= n <= 58

## Code Snippets

**C++:**

```
class Solution {
public:
int integerBreak(int n) {

}
};
```

**Java:**

```
class Solution {
public int integerBreak(int n) {

}
}
```

**Python3:**

```
class Solution:
def integerBreak(self, n: int) -> int:
```

**Python:**

```
class Solution(object):
def integerBreak(self, n):
"""
:type n: int
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number} n
* @return {number}
*/
var integerBreak = function(n) {

};
```

**TypeScript:**

```
function integerBreak(n: number): number {

};
```

**C#:**

```
public class Solution {
public int IntegerBreak(int n) {
```

```
}
}
```

**C:**

```
int integerBreak(int n) {

}
```

**Go:**

```
func integerBreak(n int) int {

}
```

**Kotlin:**

```
class Solution {
fun integerBreak(n: Int): Int {

}
}
```

**Swift:**

```
class Solution {
func integerBreak(_ n: Int) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn integer_break(n: i32) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer} n
# @return {Integer}
def integer_break(n)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function integerBreak($n) {

}
}
```

**Dart:**

```
class Solution {
int integerBreak(int n) {

}
}
```

**Scala:**

```
object Solution {
def integerBreak(n: Int): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec integer_break(n :: integer) :: integer
def integer_break(n) do

end
end
```

### Erlang:

```
-spec integer_break(N :: integer()) -> integer().
integer_break(N) ->
.
```

### Racket:

```
(define/contract (integer-break n)
(-> exact-integer? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Integer Break
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int integerBreak(int n) {

}
};
```

### Java Solution:

```
/**
* Problem: Integer Break
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
```

```
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int integerBreak(int n) {

}
}
```

### Python3 Solution:

```
"""
Problem: Integer Break
Difficulty: Medium
Tags: dp, math

Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def integerBreak(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def integerBreak(self, n):
"""
:type n: int
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Integer Break
* Difficulty: Medium
```

```
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} n
* @return {number}
*/
var integerBreak = function(n) {

};
```

### TypeScript Solution:

```
/**
* Problem: Integer Break
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function integerBreak(n: number): number {

};
```

### C# Solution:

```
/*
* Problem: Integer Break
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

public class Solution {
public int IntegerBreak(int n) {

}
}
```

### C Solution:

```
/*
* Problem: Integer Break
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

int integerBreak(int n) {

}
```

### Go Solution:

```
// Problem: Integer Break
// Difficulty: Medium
// Tags: dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func integerBreak(n int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun integerBreak(n: Int): Int {

}
}
```

### Swift Solution:

```
class Solution {
func integerBreak(_ n: Int) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Integer Break
// Difficulty: Medium
// Tags: dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn integer_break(n: i32) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def integer_break(n)

end
```

### PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $n
* @return Integer
*/
function integerBreak($n) {

}
}
```

**Dart Solution:**

```
class Solution {
int integerBreak(int n) {

}
}
```

**Scala Solution:**

```
object Solution {
def integerBreak(n: Int): Int = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec integer_break(n :: integer) :: integer
def integer_break(n) do

end
end
```

**Erlang Solution:**

```
-spec integer_break(N :: integer()) -> integer().
integer_break(N) ->
.
```

**Racket Solution:**

```
(define/contract (integer-break n)
(-> exact-integer? exact-integer?)
)
```