

Problem 1017: Convert to Base -2

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given an integer

n

, return

a binary string representing its representation in base

-2

.

Note

that the returned string should not have leading zeros unless the string is

"0"

.

Example 1:

Input:

$n = 2$

Output:

"110"

Explantion:

(-2)

2

+ (-2)

1

= 2

Example 2:

Input:

n = 3

Output:

"111"

Explantion:

(-2)

2

+ (-2)

1

+ (-2)

0

= 3

Example 3:

Input:

n = 4

Output:

"100"

Explantion:

(-2)

2

= 4

Constraints:

$0 \leq n \leq 10$

9

Code Snippets

C++:

```
class Solution {  
public:  
    string baseNeg2(int n) {  
  
    }  
};
```

Java:

```

class Solution {
public String baseNeg2(int n) {

}

}

```

Python3:

```

class Solution:
def baseNeg2(self, n: int) -> str:

```

Python:

```

class Solution(object):
def baseNeg2(self, n):
"""
:type n: int
:rtype: str
"""

```

JavaScript:

```

/**
 * @param {number} n
 * @return {string}
 */
var baseNeg2 = function(n) {

};

```

TypeScript:

```

function baseNeg2(n: number): string {

};

```

C#:

```

public class Solution {
public string BaseNeg2(int n) {

}

}

```

C:

```
char* baseNeg2(int n) {  
  
}
```

Go:

```
func baseNeg2(n int) string {  
  
}
```

Kotlin:

```
class Solution {  
    fun baseNeg2(n: Int): String {  
  
    }  
}
```

Swift:

```
class Solution {  
    func baseNeg2(_ n: Int) -> String {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn base_neg2(n: i32) -> String {  
  
    }  
}
```

Ruby:

```
# @param {Integer} n  
# @return {String}  
def base_neg2(n)  
  
end
```

PHP:

```
class Solution {

    /**
     * @param Integer $n
     * @return String
     */
    function baseNeg2($n) {

    }

}
```

Dart:

```
class Solution {
  String baseNeg2(int n) {

  }

}
```

Scala:

```
object Solution {
  def baseNeg2(n: Int): String = {

  }

}
```

Elixir:

```
defmodule Solution do
  @spec base_neg2(n :: integer) :: String.t
  def base_neg2(n) do

  end

end
```

Erlang:

```
-spec base_neg2(N :: integer()) -> unicode:unicode_binary().
base_neg2(N) ->
.
```

Racket:

```
(define/contract (base-neg2 n)
  (-> exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Convert to Base -2
 * Difficulty: Medium
 * Tags: string, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    string baseNeg2(int n) {

    }
};
```

Java Solution:

```
/**
 * Problem: Convert to Base -2
 * Difficulty: Medium
 * Tags: string, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public String baseNeg2(int n) {
```

```
}  
}
```

Python3 Solution:

```
"""  
Problem: Convert to Base -2  
Difficulty: Medium  
Tags: string, math  
  
Approach: String manipulation with hash map or two pointers  
Time Complexity: O(n) or O(n log n)  
Space Complexity: O(1) to O(n) depending on approach  
"""  
  
class Solution:  
    def baseNeg2(self, n: int) -> str:  
        # TODO: Implement optimized solution  
        pass
```

Python Solution:

```
class Solution(object):  
    def baseNeg2(self, n):  
        """  
        :type n: int  
        :rtype: str  
        """
```

JavaScript Solution:

```
/**  
 * Problem: Convert to Base -2  
 * Difficulty: Medium  
 * Tags: string, math  
 *  
 * Approach: String manipulation with hash map or two pointers  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */
```

```

/**
 * @param {number} n
 * @return {string}
 */
var baseNeg2 = function(n) {

};

```

TypeScript Solution:

```

/**
 * Problem: Convert to Base -2
 * Difficulty: Medium
 * Tags: string, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function baseNeg2(n: number): string {

};

```

C# Solution:

```

/*
 * Problem: Convert to Base -2
 * Difficulty: Medium
 * Tags: string, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public string BaseNeg2(int n) {

    }
}

```

```
}
```

C Solution:

```
/*
 * Problem: Convert to Base -2
 * Difficulty: Medium
 * Tags: string, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

char* baseNeg2(int n) {

}
```

Go Solution:

```
// Problem: Convert to Base -2
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func baseNeg2(n int) string {

}
```

Kotlin Solution:

```
class Solution {
    fun baseNeg2(n: Int): String {

    }
}
```

Swift Solution:

```

class Solution {
    func baseNeg2(_ n: Int) -> String {

    }
}

```

Rust Solution:

```

// Problem: Convert to Base -2
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn base_neg2(n: i32) -> String {

    }
}

```

Ruby Solution:

```

# @param {Integer} n
# @return {String}
def base_neg2(n)

end

```

PHP Solution:

```

class Solution {

    /**
     * @param Integer $n
     * @return String
     */
    function baseNeg2($n) {

    }
}

```

Dart Solution:

```
class Solution {  
  String baseNeg2(int n) {  
  
  }  
}
```

Scala Solution:

```
object Solution {  
  def baseNeg2(n: Int): String = {  
  
  }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec base_neg2(n :: integer) :: String.t  
  def base_neg2(n) do  
  
  end  
end
```

Erlang Solution:

```
-spec base_neg2(N :: integer()) -> unicode:unicode_binary().  
base_neg2(N) ->  
.
```

Racket Solution:

```
(define/contract (base-neg2 n)  
  (-> exact-integer? string?)  
)
```