Problem 7: Reverse Integer

Problem Information

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

Problem Description

Given a signed 32-bit integer

x

, return

x

with its digits reversed

. If reversing

x

causes the value to go outside the signed 32-bit integer range

[-2

31

, 2

31

- 1]

, then return

0

.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

Input:

x = 123

Output:

321

Example 2:

Input:

x = -123

Output:

-321

Example 3:

Input:

x = 120

Output:

21

Constraints:

-2

31

<= x <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
int reverse(int x) {

}
};
```

Java:

```
class Solution {
public int reverse(int x) {

}
}
```

Python3:

```
class Solution:
def reverse(self, x: int) -> int:
```

Python:

```
class Solution(object):
def reverse(self, x):
"""
:type x: int
```

```
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} x
* @return {number}
*/
var reverse = function(x) {

};
```

TypeScript:

```
function reverse(x: number): number {

};
```

C#:

```
public class Solution {
public int Reverse(int x) {

}
}
```

C:

```
int reverse(int x){

}
```

Go:

```
func reverse(x int) int {

}
```

Kotlin:

```
class Solution {
fun reverse(x: Int): Int {

}
}
```

Swift:

```
class Solution {
func reverse(_ x: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn reverse(x: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} x
# @return {Integer}
def reverse(x)

end
```

PHP:

```
class Solution {

/**
* @param Integer $x
* @return Integer
*/
function reverse($x) {

}
}
```

Dart:

```
class Solution {
int reverse(int x) {

}
}
```

Scala:

```
object Solution {
def reverse(x: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec reverse(x :: integer) :: integer
def reverse(x) do

end
end
```

Erlang:

```
-spec reverse(X :: integer()) -> integer().
reverse(X) ->
.
```

Racket:

```
(define/contract (reverse x)
(-> exact-integer? exact-integer?)

)
```

Solutions

C++ Solution:

```
/*
* Problem: Reverse Integer
* Difficulty: Medium
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int reverse(int x) {

}
};
```

Java Solution:

```
/**
* Problem: Reverse Integer
* Difficulty: Medium
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int reverse(int x) {

}
}
```

Python3 Solution:

```
"""
Problem: Reverse Integer
Difficulty: Medium
Tags: math
```

```
Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

```
class Solution(object):
def reverse(self, x):
"""
:type x: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Reverse Integer
* Difficulty: Medium
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} x
* @return {number}
*/
var reverse = function(x) {

};
```

TypeScript Solution:

```
/**
* Problem: Reverse Integer
* Difficulty: Medium
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function reverse(x: number): number {

};
```

C# Solution:

```
/*
* Problem: Reverse Integer
* Difficulty: Medium
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int Reverse(int x) {

}
}
```

C Solution:

```
/*
* Problem: Reverse Integer
* Difficulty: Medium
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

int reverse(int x){

}
```

Go Solution:

```
// Problem: Reverse Integer
// Difficulty: Medium
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func reverse(x int) int {

}
```

Kotlin Solution:

```
class Solution {
fun reverse(x: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func reverse(_ x: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Reverse Integer
// Difficulty: Medium
// Tags: math
```

```
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn reverse(x: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} x
# @return {Integer}
def reverse(x)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $x
* @return Integer
*/
function reverse($x) {

}
}
```

Dart Solution:

```
class Solution {
int reverse(int x) {

}
}
```

Scala Solution:

```
object Solution {
def reverse(x: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec reverse(x :: integer) :: integer
def reverse(x) do

end
end
```

Erlang Solution:

```
-spec reverse(X :: integer()) -> integer().
reverse(X) ->
.
```

Racket Solution:

```
(define/contract (reverse x)
(-> exact-integer? exact-integer?)

)
```