Problem 2591: Distribute Money to Maximum Children

Problem Information

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

Problem Description

You are given an integer

money

denoting the amount of money (in dollars) that you have and another integer

children

denoting the number of children that you must distribute the money to.

You have to distribute the money according to the following rules:

All money must be distributed.

Everyone must receive at least

1

dollar.

Nobody receives

4

dollars.

Return

the

maximum

number of children who may receive

exactly

8

dollars if you distribute the money according to the aforementioned rules

. If there is no way to distribute the money, return

-1

.

Example 1:

Input:

money = 20, children = 3

Output:

1

Explanation:

The maximum number of children with 8 dollars will be 1. One of the ways to distribute the money is: - 8 dollars to the first child. - 9 dollars to the second child. - 3 dollars to the third child. It can be proven that no distribution exists such that number of children getting 8 dollars is greater than 1.

Example 2:

Input:

money = 16, children = 2

Output:

2

Explanation:

Each child can be given 8 dollars.

Constraints:

1 <= money <= 200

2 <= children <= 30

Code Snippets

C++:

```
class Solution {
public:
int distMoney(int money, int children) {

}
};
```

Java:

```
class Solution {
public int distMoney(int money, int children) {

}
}
```

Python3:

```
class Solution:
def distMoney(self, money: int, children: int) -> int:
```

Python:

```
class Solution(object):
def distMoney(self, money, children):
"""
:type money: int
:type children: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} money
* @param {number} children
* @return {number}
*/
var distMoney = function(money, children) {

};
```

TypeScript:

```
function distMoney(money: number, children: number): number {

};
```

C#:

```
public class Solution {
public int DistMoney(int money, int children) {

}
}
```

C:

```
int distMoney(int money, int children) {

}
```

Go:

```
func distMoney(money int, children int) int {

}
```

Kotlin:

```
class Solution {
fun distMoney(money: Int, children: Int): Int {

}
}
```

Swift:

```
class Solution {
func distMoney(_ money: Int, _ children: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn dist_money(money: i32, children: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} money
# @param {Integer} children
# @return {Integer}
def dist_money(money, children)

end
```

PHP:

```
class Solution {
```

```
/**
* @param Integer $money
* @param Integer $children
* @return Integer
*/
function distMoney($money, $children) {

}
}
```

Dart:

```
class Solution {
int distMoney(int money, int children) {

}
}
```

Scala:

```
object Solution {
def distMoney(money: Int, children: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec dist_money(money :: integer, children :: integer) :: integer
def dist_money(money, children) do

end
end
```

Erlang:

```
-spec dist_money(Money :: integer(), Children :: integer()) -> integer().
dist_money(Money, Children) ->
.
```

Racket:

```
(define/contract (dist-money money children)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Distribute Money to Maximum Children
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 distMoney(int money, int children) {

}
};
```

Java Solution:

```
/**
* Problem: Distribute Money to Maximum Children
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 distMoney(int money, int children) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Distribute Money to Maximum Children
Difficulty: Easy
Tags: greedy, math

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def distMoney(self, money: int, children: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def distMoney(self, money, children):
"""
:type money: int
:type children: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Distribute Money to Maximum Children
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
/**
* @param {number} money
* @param {number} children
* @return {number}
*/
var distMoney = function(money, children) {

};
```

TypeScript Solution:

```
/**
* Problem: Distribute Money to Maximum Children
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function distMoney(money: number, children: number): number {

};
```

C# Solution:

```
/*
* Problem: Distribute Money to Maximum Children
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 DistMoney(int money, int children) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Distribute Money to Maximum Children
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int distMoney(int money, int children) {

}
```

Go Solution:

```
// Problem: Distribute Money to Maximum Children
// Difficulty: Easy
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func distMoney(money int, children int) int {

}
```

Kotlin Solution:

```
class Solution {
fun distMoney(money: Int, children: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func distMoney(_ money: Int, _ children: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Distribute Money to Maximum Children
// Difficulty: Easy
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// 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 dist_money(money: i32, children: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} money
# @param {Integer} children
# @return {Integer}
def dist_money(money, children)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $money
* @param Integer $children
* @return Integer
*/
function distMoney($money, $children) {
```

```
}
}
```

Dart Solution:

```
class Solution {
int distMoney(int money, int children) {

}
}
```

Scala Solution:

```
object Solution {
def distMoney(money: Int, children: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec dist_money(money :: integer, children :: integer) :: integer
def dist_money(money, children) do

end
end
```

Erlang Solution:

```
-spec dist_money(Money :: integer(), Children :: integer()) -> integer().
dist_money(Money, Children) ->
.
```

Racket Solution:

```
(define/contract (dist-money money children)
(-> exact-integer? exact-integer? exact-integer?)
)
```