Problem 2469: Convert the Temperature

Problem Information

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

Problem Description

You are given a non-negative floating point number rounded to two decimal places

celsius

, that denotes the

temperature in Celsius

.

You should convert Celsius into

Kelvin

and

Fahrenheit

and return it as an array

ans = [kelvin, fahrenheit]

.

Return

the array

ans

.

Answers within

10

-5

of the actual answer will be accepted.

Note that:

Kelvin = Celsius + 273.15

Fahrenheit = Celsius * 1.80 + 32.00

Example 1:

Input:

celsius = 36.50

Output:

[309.65000,97.70000]

Explanation:

Temperature at 36.50 Celsius converted in Kelvin is 309.65 and converted in Fahrenheit is 97.70.

Example 2:

Input:

celsius = 122.11

Output:

[395.26000,251.79800]

Explanation:

Temperature at 122.11 Celsius converted in Kelvin is 395.26 and converted in Fahrenheit is 251.798.

Constraints:

0 <= celsius <= 1000

Code Snippets

C++:

```
class Solution {
public:
vector<double> convertTemperature(double celsius) {

}
};
```

Java:

```
class Solution {
public double[] convertTemperature(double celsius) {

}
}
```

Python3:

```
class Solution:
def convertTemperature(self, celsius: float) -> List[float]:
```

Python:

```
class Solution(object):
def convertTemperature(self, celsius):
"""
:type celsius: float
:rtype: List[float]
"""
```

JavaScript:

```
/**
* @param {number} celsius
* @return {number[]}
*/
var convertTemperature = function(celsius) {

};
```

TypeScript:

```
function convertTemperature(celsius: number): number[] {

};
```

C#:

```
public class Solution {
public double[] ConvertTemperature(double celsius) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
double* convertTemperature(double celsius, int* returnSize) {

}
```

Go:

```
func convertTemperature(celsius float64) []float64 {

}
```

Kotlin:

```
class Solution {
fun convertTemperature(celsius: Double): DoubleArray {

}
}
```

Swift:

```
class Solution {
func convertTemperature(_ celsius: Double) -> [Double] {

}
}
```

Rust:

```
impl Solution {
pub fn convert_temperature(celsius: f64) -> Vec<f64> {

}
}
```

Ruby:

```
# @param {Float} celsius
# @return {Float[]}
def convert_temperature(celsius)

end
```

PHP:

```
class Solution {

/**
* @param Float $celsius
* @return Float[]
```

```
*/
function convertTemperature($celsius) {

}
}
```

Dart:

```
class Solution {
List<double> convertTemperature(double celsius) {

}
}
```

Scala:

```
object Solution {
def convertTemperature(celsius: Double): Array[Double] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec convert_temperature(celsius :: float) :: [float]
def convert_temperature(celsius) do

end
end
```

Erlang:

```
-spec convert_temperature(Celsius :: float()) -> [float()].
convert_temperature(Celsius) ->
.
```

Racket:

```
(define/contract (convert-temperature celsius)
(-> flonum? (listof flonum?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Convert the Temperature
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<double> convertTemperature(double celsius) {

}
};
```

Java Solution:

```
/**
* Problem: Convert the Temperature
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public double[] convertTemperature(double celsius) {

}
}
```

Python3 Solution:

```
"""
Problem: Convert the Temperature
Difficulty: Easy
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def convertTemperature(self, celsius: float) -> List[float]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def convertTemperature(self, celsius):
"""
:type celsius: float
:rtype: List[float]
"""
```

JavaScript Solution:

```
/**
* Problem: Convert the Temperature
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} celsius
* @return {number[]}
*/
var convertTemperature = function(celsius) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Convert the Temperature
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function convertTemperature(celsius: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Convert the Temperature
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public double[] ConvertTemperature(double celsius) {

}
}
```

C Solution:

```
/*
* Problem: Convert the Temperature
* Difficulty: Easy
```

```
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
double* convertTemperature(double celsius, int* returnSize) {

}
```

Go Solution:

```
// Problem: Convert the Temperature
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func convertTemperature(celsius float64) []float64 {

}
```

Kotlin Solution:

```
class Solution {
fun convertTemperature(celsius: Double): DoubleArray {

}
}
```

Swift Solution:

```
class Solution {
func convertTemperature(_ celsius: Double) -> [Double] {
```

```
}
}
```

Rust Solution:

```
// Problem: Convert the Temperature
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn convert_temperature(celsius: f64) -> Vec<f64> {

}
}
```

Ruby Solution:

```
# @param {Float} celsius
# @return {Float[]}
def convert_temperature(celsius)

end
```

PHP Solution:

```
class Solution {

/**
* @param Float $celsius
* @return Float[]
*/
function convertTemperature($celsius) {

}
}
```

Dart Solution:

```
class Solution {
List<double> convertTemperature(double celsius) {

}
}
```

Scala Solution:

```
object Solution {
def convertTemperature(celsius: Double): Array[Double] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec convert_temperature(celsius :: float) :: [float]
def convert_temperature(celsius) do

end
end
```

Erlang Solution:

```
-spec convert_temperature(Celsius :: float()) -> [float()].
convert_temperature(Celsius) ->
.
```

Racket Solution:

```
(define/contract (convert-temperature celsius)
(-> flonum? (listof flonum?))
)
```