Problem 1344: Angle Between Hands of a Clock

Problem Information

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

Problem Description

Given two numbers,

hour

and

minutes

, return

the smaller angle (in degrees) formed between the

hour

and the

minute

hand

.

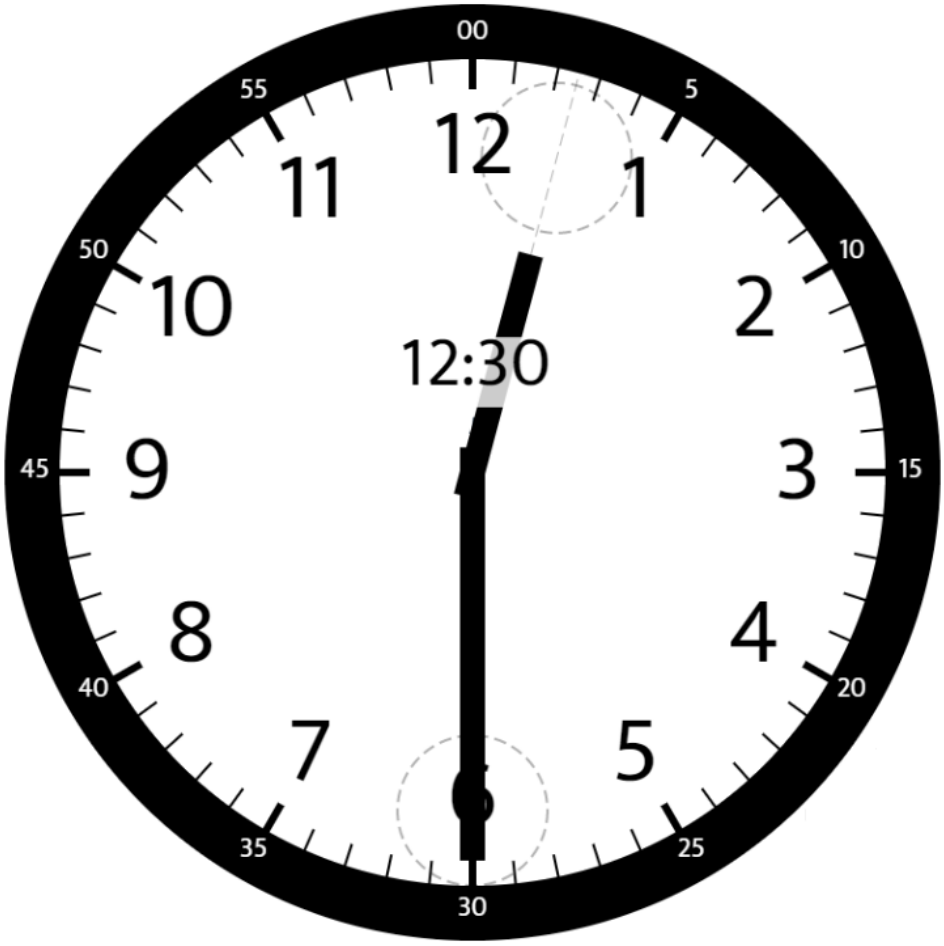
Answers within

10

-5

of the actual value will be accepted as correct.

Example 1:



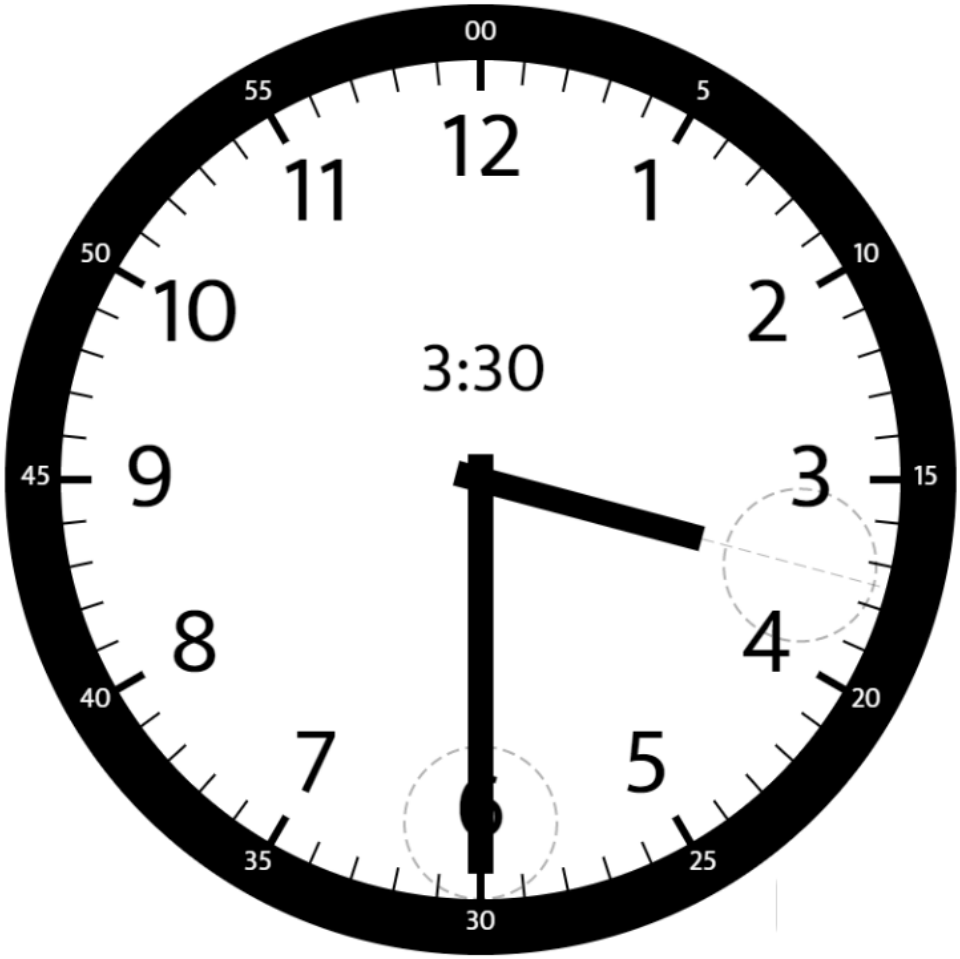
Input:

hour = 12, minutes = 30

Output:

165

Example 2:



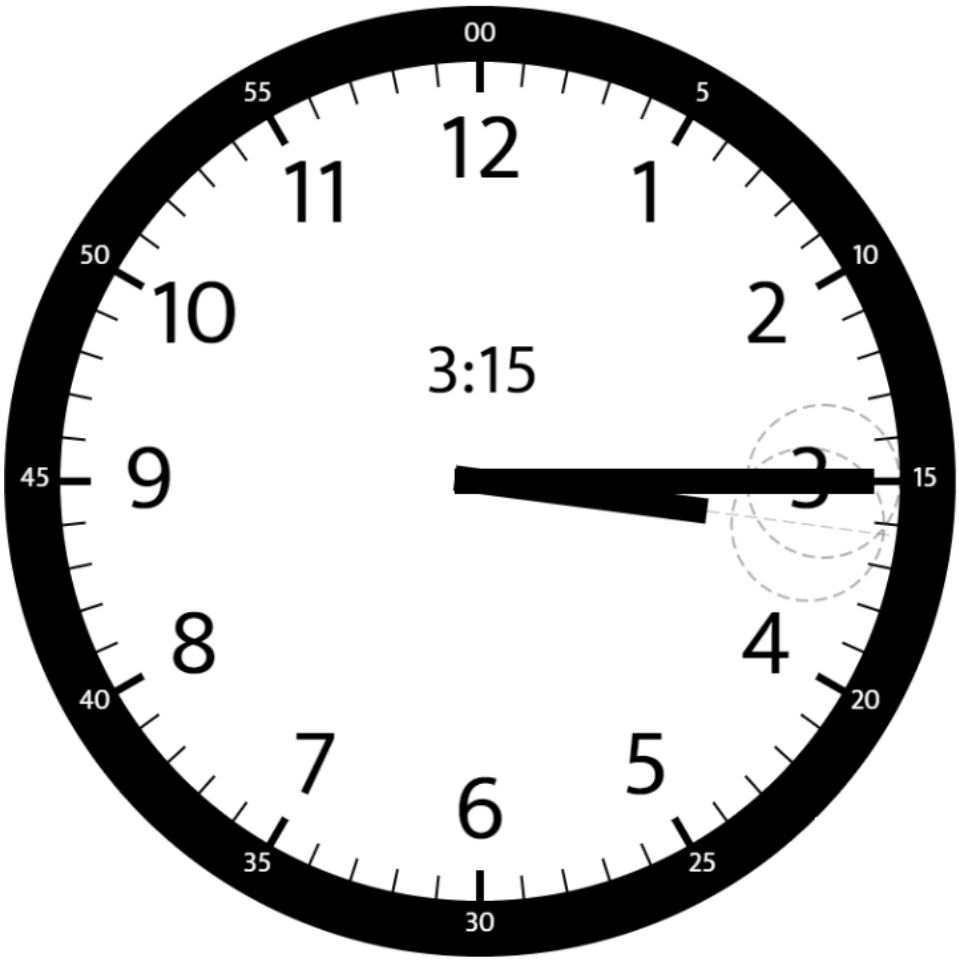
Input:

hour = 3, minutes = 30

Output:

75

Example 3:



Input:

hour = 3, minutes = 15

Output:

7.5

Constraints:

1 <= hour <= 12

0 <= minutes <= 59

Code Snippets

C++:

```
class Solution {
public:
double angleClock(int hour, int minutes) {

}
};
```

Java:

```
class Solution {
public double angleClock(int hour, int minutes) {

}
}
```

Python3:

```
class Solution:
def angleClock(self, hour: int, minutes: int) -> float:
```

Python:

```
class Solution(object):
def angleClock(self, hour, minutes):
"""
:type hour: int
:type minutes: int
:rtype: float
"""
```

JavaScript:

```
/**
* @param {number} hour
* @param {number} minutes
* @return {number}
*/
var angleClock = function(hour, minutes) {

};
```

TypeScript:

```
function angleClock(hour: number, minutes: number): number {

};
```

C#:

```
public class Solution {
public double AngleClock(int hour, int minutes) {

}
}
```

C:

```
double angleClock(int hour, int minutes) {

}
```

Go:

```
func angleClock(hour int, minutes int) float64 {

}
```

Kotlin:

```
class Solution {
fun angleClock(hour: Int, minutes: Int): Double {

}
}
```

Swift:

```
class Solution {
func angleClock(_ hour: Int, _ minutes: Int) -> Double {

}
}
```

Rust:

```
impl Solution {
pub fn angle_clock(hour: i32, minutes: i32) -> f64 {

}
}
```

Ruby:

```
# @param {Integer} hour
# @param {Integer} minutes
# @return {Float}
def angle_clock(hour, minutes)

end
```

PHP:

```
class Solution {

/**
* @param Integer $hour
* @param Integer $minutes
* @return Float
*/
function angleClock($hour, $minutes) {

}
}
```

Dart:

```
class Solution {
double angleClock(int hour, int minutes) {

}
}
```

Scala:

```
object Solution {
def angleClock(hour: Int, minutes: Int): Double = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec angle_clock(hour :: integer, minutes :: integer) :: float
def angle_clock(hour, minutes) do

end
end
```

Erlang:

```
-spec angle_clock(Hour :: integer(), Minutes :: integer()) -> float().
angle_clock(Hour, Minutes) ->
.
```

Racket:

```
(define/contract (angle-clock hour minutes)
(-> exact-integer? exact-integer? flonum?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Angle Between Hands of a Clock
* 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:
double angleClock(int hour, int minutes) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Angle Between Hands of a Clock
* 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 double angleClock(int hour, int minutes) {

}
}
```

Python3 Solution:

```
"""
Problem: Angle Between Hands of a Clock
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 angleClock(self, hour: int, minutes: int) -> float:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def angleClock(self, hour, minutes):
"""
:type hour: int
:type minutes: int
:rtype: float
"""
```

JavaScript Solution:

```
/**
* Problem: Angle Between Hands of a Clock
* 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} hour
* @param {number} minutes
* @return {number}
*/
var angleClock = function(hour, minutes) {

};
```

TypeScript Solution:

```
/**
* Problem: Angle Between Hands of a Clock
* 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 angleClock(hour: number, minutes: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Angle Between Hands of a Clock
* 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 double AngleClock(int hour, int minutes) {

}
}
```

C Solution:

```
/*
* Problem: Angle Between Hands of a Clock
* 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
*/

double angleClock(int hour, int minutes) {

}
```

Go Solution:

```
// Problem: Angle Between Hands of a Clock
// 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 angleClock(hour int, minutes int) float64 {

}
```

Kotlin Solution:

```
class Solution {
fun angleClock(hour: Int, minutes: Int): Double {

}
}
```

Swift Solution:

```
class Solution {
func angleClock(_ hour: Int, _ minutes: Int) -> Double {

}
}
```

Rust Solution:

```
// Problem: Angle Between Hands of a Clock
// 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 angle_clock(hour: i32, minutes: i32) -> f64 {

}
}
```

Ruby Solution:

```
# @param {Integer} hour
# @param {Integer} minutes
# @return {Float}
def angle_clock(hour, minutes)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $hour
* @param Integer $minutes
* @return Float
*/
function angleClock($hour, $minutes) {

}
}
```

Dart Solution:

```
class Solution {
double angleClock(int hour, int minutes) {

}
}
```

Scala Solution:

```
object Solution {
def angleClock(hour: Int, minutes: Int): Double = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec angle_clock(hour :: integer, minutes :: integer) :: float
def angle_clock(hour, minutes) do

end
end
```

Erlang Solution:

```
-spec angle_clock(Hour :: integer(), Minutes :: integer()) -> float().
angle_clock(Hour, Minutes) ->
.
```

Racket Solution:

```
(define/contract (angle-clock hour minutes)
(-> exact-integer? exact-integer? flonum?)
)
```