Problem 1396: Design Underground System

Problem Information

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

Problem Description

An underground railway system is keeping track of customer travel times between different stations. They are using this data to calculate the average time it takes to travel from one station to another.

Implement the

UndergroundSystem

class:

void checkIn(int id, string stationName, int t)

A customer with a card ID equal to

id

, checks in at the station

stationName

at time

t

.

A customer can only be checked into one place at a time.

void checkOut(int id, string stationName, int t)

A customer with a card ID equal to

id

, checks out from the station

stationName

at time

t

.

double getAverageTime(string startStation, string endStation)

Returns the average time it takes to travel from

startStation

to

endStation

.

The average time is computed from all the previous traveling times from

startStation

to

endStation

that happened

directly

, meaning a check in at

startStation

followed by a check out from

endStation

.

The time it takes to travel from

startStation

to

endStation

may be different

from the time it takes to travel from

endStation

to

startStation

.

There will be at least one customer that has traveled from

startStation

to

endStation

before

getAverageTime

is called.

You may assume all calls to the

checkIn

and

checkOut

methods are consistent. If a customer checks in at time

t

1

then checks out at time

t

2

, then

t

1

< t

2

. All events happen in chronological order.

Example 1:

Input

["UndergroundSystem","checkIn","checkIn","checkIn","checkOut","checkOut","checkOut","getAverageTime","getAverageTime","checkIn","getAverageTime","checkOut","getAverageTime"] [[],[45,"Leyton",3],[32,"Paradise",8],[27,"Leyton",10],[45,"Waterloo",15],[27,"Waterloo",20],[32,"Cambridge",22],["Paradise","Cambridge"],["Leyton","Waterloo"],[10,"Leyton",24],["Leyton","Waterloo"],[10,"Waterloo",38],["Leyton","Waterloo"]]

Output

[null,null,null,null,null,null,null,14.00000,11.00000,null,11.00000,null,12.00000]

Explanation

UndergroundSystem undergroundSystem = new UndergroundSystem(); undergroundSystem.checkIn(45, "Leyton", 3); undergroundSystem.checkIn(32, "Paradise", 8); undergroundSystem.checkIn(27, "Leyton", 10); undergroundSystem.checkOut(45, "Waterloo", 15); // Customer 45 "Leyton" -> "Waterloo" in 15-3 = 12 undergroundSystem.checkOut(27, "Waterloo", 20); // Customer 27 "Leyton" -> "Waterloo" in 20-10 = 10 undergroundSystem.checkOut(32, "Cambridge", 22); // Customer 32 "Paradise" -> "Cambridge" in 22-8 = 14 undergroundSystem.getAverageTime("Paradise", "Cambridge"); // return 14.00000. One trip "Paradise" -> "Cambridge", (14) / 1 = 14 undergroundSystem.getAverageTime("Leyton", "Waterloo"); // return 11.00000. Two trips "Leyton" -> "Waterloo", (10 + 12) / 2 = 11 undergroundSystem.checkIn(10, "Leyton", 24); undergroundSystem.getAverageTime("Leyton", "Waterloo"); // return 11.00000 undergroundSystem.checkOut(10, "Waterloo", 38); // Customer 10 "Leyton" -> "Waterloo" in 38-24 = 14 undergroundSystem.getAverageTime("Leyton", "Waterloo"); // return 12.00000. Three trips "Leyton" -> "Waterloo", (10 + 12 + 14) / 3 = 12

Example 2:

Input

["UndergroundSystem","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime"] [[],[10,"Leyton",3],[10,"Paradise",8],["Leyton","Paradise"],[5,"Leyton",10],[5,"Paradise",16],["Leyton","Paradise"],[2,"Leyton",21],[2,"Paradise",30],["Leyton","Paradise"]]

Output

[null,null,null,5.00000,null,null,5.50000,null,null,6.66667]

Explanation

UndergroundSystem undergroundSystem = new UndergroundSystem();
undergroundSystem.checkIn(10, "Leyton", 3); undergroundSystem.checkOut(10, "Paradise", 8); // Customer 10 "Leyton" -> "Paradise" in 8-3 = 5
undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.00000, (5) / 1 = 5
undergroundSystem.checkIn(5, "Leyton", 10); undergroundSystem.checkOut(5, "Paradise", 16); // Customer 5 "Leyton" -> "Paradise" in 16-10 = 6
undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.50000, (5 + 6) / 2 = 5.5
undergroundSystem.checkIn(2, "Leyton", 21); undergroundSystem.checkOut(2, "Paradise", 30); // Customer 2 "Leyton" -> "Paradise" in 30-21 = 9
undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 6.66667, (5 + 6 + 9) / 3 = 6.66667

Constraints:

1 <= id, t <= 10

6

1 <= stationName.length, startStation.length, endStation.length <= 10

All strings consist of uppercase and lowercase English letters and digits.

There will be at most

2 * 10

4

calls

in total

to

checkIn

,

checkOut

, and

getAverageTime

.

Answers within

10

-5

of the actual value will be accepted.

Code Snippets

C++:

```
class UndergroundSystem {
public:
UndergroundSystem() {

}

void checkIn(int id, string stationName, int t) {

}

void checkOut(int id, string stationName, int t) {

}

double getAverageTime(string startStation, string endStation) {
```

```
}
};

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem* obj = new UndergroundSystem();
* obj->checkIn(id,stationName,t);
* obj->checkOut(id,stationName,t);
* double param_3 = obj->getAverageTime(startStation,endStation);
*/
```

Java:

```
class UndergroundSystem {

public UndergroundSystem() {

}

public void checkIn(int id, String stationName, int t) {

}

public void checkOut(int id, String stationName, int t) {

}

public double getAverageTime(String startStation, String endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem obj = new UndergroundSystem();
* obj.checkIn(id,stationName,t);
* obj.checkOut(id,stationName,t);
* double param_3 = obj.getAverageTime(startStation,endStation);
*/
```

Python3:

```
class UndergroundSystem:

def __init__(self):


def checkIn(self, id: int, stationName: str, t: int) -> None:


def checkOut(self, id: int, stationName: str, t: int) -> None:


def getAverageTime(self, startStation: str, endStation: str) -> float:



# Your UndergroundSystem object will be instantiated and called as such:
# obj = UndergroundSystem()
# obj.checkIn(id,stationName,t)
# obj.checkOut(id,stationName,t)
# param_3 = obj.getAverageTime(startStation,endStation)
```

Python:

```
class UndergroundSystem(object):

def __init__(self):


def checkIn(self, id, stationName, t):
"""
:type id: int
:type stationName: str
:type t: int
:rtype: None
"""


def checkOut(self, id, stationName, t):
"""
:type id: int
:type stationName: str
```

```
:type t: int
:rtype: None
"""


def getAverageTime(self, startStation, endStation):
"""
:type startStation: str
:type endStation: str
:rtype: float
"""



# Your UndergroundSystem object will be instantiated and called as such:
# obj = UndergroundSystem()
# obj.checkIn(id,stationName,t)
# obj.checkOut(id,stationName,t)
# param_3 = obj.getAverageTime(startStation,endStation)
```

JavaScript:

```
var UndergroundSystem = function() {

};

/**
* @param {number} id
* @param {string} stationName
* @param {number} t
* @return {void}
*/
UndergroundSystem.prototype.checkIn = function(id, stationName, t) {

};

/**
* @param {number} id
* @param {string} stationName
* @param {number} t
* @return {void}
```

```
*/
UndergroundSystem.prototype.checkOut = function(id, stationName, t) {

};

/**
* @param {string} startStation
* @param {string} endStation
* @return {number}
*/
UndergroundSystem.prototype.getAverageTime = function(startStation,
endStation) {

};

/**
* Your UndergroundSystem object will be instantiated and called as such:
* var obj = new UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* var param_3 = obj.getAverageTime(startStation,endStation)
*/
```

TypeScript:

```
class UndergroundSystem {
constructor() {

}

checkIn(id: number, stationName: string, t: number): void {

}

checkOut(id: number, stationName: string, t: number): void {

}

getAverageTime(startStation: string, endStation: string): number {

}
}
```

```
/**
* Your UndergroundSystem object will be instantiated and called as such:
* var obj = new UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* var param_3 = obj.getAverageTime(startStation,endStation)
*/
```

C#:

```
public class UndergroundSystem {

public UndergroundSystem() {

}

public void CheckIn(int id, string stationName, int t) {

}

public void CheckOut(int id, string stationName, int t) {

}

public double GetAverageTime(string startStation, string endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem obj = new UndergroundSystem();
* obj.CheckIn(id,stationName,t);
* obj.CheckOut(id,stationName,t);
* double param_3 = obj.GetAverageTime(startStation,endStation);
*/
```

C:

```
```

```
typedef struct {

} UndergroundSystem;


UndergroundSystem* undergroundSystemCreate() {

}

void undergroundSystemCheckIn(UndergroundSystem* obj, int id, char*
stationName, int t) {

}

void undergroundSystemCheckOut(UndergroundSystem* obj, int id, char*
stationName, int t) {

}

double undergroundSystemGetAverageTime(UndergroundSystem* obj, char*
startStation, char* endStation) {

}

void undergroundSystemFree(UndergroundSystem* obj) {

}

/**
* Your UndergroundSystem struct will be instantiated and called as such:
* UndergroundSystem* obj = undergroundSystemCreate();
* undergroundSystemCheckIn(obj, id, stationName, t);

* undergroundSystemCheckOut(obj, id, stationName, t);

* double param_3 = undergroundSystemGetAverageTime(obj, startStation,
endStation);

* undergroundSystemFree(obj);
*/
```

Go:

```
type UndergroundSystem struct {

}


func Constructor() UndergroundSystem {

}


func (this *UndergroundSystem) CheckIn(id int, stationName string, t int) {

}


func (this *UndergroundSystem) CheckOut(id int, stationName string, t int) {

}


func (this *UndergroundSystem) GetAverageTime(startStation string, endStation
string) float64 {

}


/**
* Your UndergroundSystem object will be instantiated and called as such:
* obj := Constructor();
* obj.CheckIn(id,stationName,t);
* obj.CheckOut(id,stationName,t);
* param_3 := obj.GetAverageTime(startStation,endStation);
*/
```

Kotlin:

```
class UndergroundSystem() {

fun checkIn(id: Int, stationName: String, t: Int) {
```

```
}

fun checkOut(id: Int, stationName: String, t: Int) {

}

fun getAverageTime(startStation: String, endStation: String): Double {

}

}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* var obj = UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* var param_3 = obj.getAverageTime(startStation,endStation)
*/
```

Swift:

```
class UndergroundSystem {

init() {

}

func checkIn(_ id: Int, _ stationName: String, _ t: Int) {

}

func checkOut(_ id: Int, _ stationName: String, _ t: Int) {

}

func getAverageTime(_ startStation: String, _ endStation: String) -> Double {

}
}
```

```
/**
* Your UndergroundSystem object will be instantiated and called as such:
* let obj = UndergroundSystem()
* obj.checkIn(id, stationName, t)
* obj.checkOut(id, stationName, t)
* let ret_3: Double = obj.getAverageTime(startStation, endStation)
*/
```

Rust:

```
struct UndergroundSystem {

}


/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl UndergroundSystem {

fn new() -> Self {

}

fn check_in(&self, id: i32, station_name: String, t: i32) {

}

fn check_out(&self, id: i32, station_name: String, t: i32) {

}

fn get_average_time(&self, start_station: String, end_station: String) -> f64
{

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* let obj = UndergroundSystem::new();
```

```
* obj.check_in(id, stationName, t);
* obj.check_out(id, stationName, t);
* let ret_3: f64 = obj.get_average_time(startStation, endStation);
*/
```

Ruby:

```
class UndergroundSystem
def initialize()

end


=begin
:type id: Integer
:type station_name: String
:type t: Integer
:rtype: Void
=end
def check_in(id, station_name, t)

end


=begin
:type id: Integer
:type station_name: String
:type t: Integer
:rtype: Void
=end
def check_out(id, station_name, t)

end


=begin
:type start_station: String
:type end_station: String
:rtype: Float
=end
def get_average_time(start_station, end_station)
```

```
end


end

# Your UndergroundSystem object will be instantiated and called as such:
# obj = UndergroundSystem.new()
# obj.check_in(id, station_name, t)
# obj.check_out(id, station_name, t)
# param_3 = obj.get_average_time(start_station, end_station)
```

PHP:

```
class UndergroundSystem {
/**
*/
function __construct() {

}

/**
* @param Integer $id
* @param String $stationName
* @param Integer $t
* @return NULL
*/
function checkIn($id, $stationName, $t) {

}

/**
* @param Integer $id
* @param String $stationName
* @param Integer $t
* @return NULL
*/
function checkOut($id, $stationName, $t) {

}

/**
* @param String $startStation
```

```
* @param String $endStation
* @return Float
*/
function getAverageTime($startStation, $endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* $obj = UndergroundSystem();
* $obj->checkIn($id, $stationName, $t);
* $obj->checkOut($id, $stationName, $t);
* $ret_3 = $obj->getAverageTime($startStation, $endStation);
*/
```

Dart:

```
class UndergroundSystem {

UndergroundSystem() {

}

void checkIn(int id, String stationName, int t) {

}

void checkOut(int id, String stationName, int t) {

}

double getAverageTime(String startStation, String endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem obj = UndergroundSystem();
* obj.checkIn(id,stationName,t);
* obj.checkOut(id,stationName,t);
```

```
* double param3 = obj.getAverageTime(startStation,endStation);
*/
```

Scala:

```
class UndergroundSystem() {

def checkIn(id: Int, stationName: String, t: Int): Unit = {

}

def checkOut(id: Int, stationName: String, t: Int): Unit = {

}

def getAverageTime(startStation: String, endStation: String): Double = {

}

}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* val obj = new UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* val param_3 = obj.getAverageTime(startStation,endStation)
*/
```

Elixir:

```
defmodule UndergroundSystem do
@spec init_() :: any
def init_() do

end

@spec check_in(id :: integer, station_name :: String.t, t :: integer) :: any
def check_in(id, station_name, t) do

end
```

```
@spec check_out(id :: integer, station_name :: String.t, t :: integer) :: any
def check_out(id, station_name, t) do

end

@spec get_average_time(start_station :: String.t, end_station :: String.t) ::
float
def get_average_time(start_station, end_station) do

end
end

# Your functions will be called as such:
# UndergroundSystem.init_()
# UndergroundSystem.check_in(id, station_name, t)
# UndergroundSystem.check_out(id, station_name, t)
# param_3 = UndergroundSystem.get_average_time(start_station, end_station)

# UndergroundSystem.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang:

```
-spec underground_system_init_() -> any().
underground_system_init_() ->
.

-spec underground_system_check_in(Id :: integer(), StationName ::
unicode:unicode_binary(), T :: integer()) -> any().
underground_system_check_in(Id, StationName, T) ->
.

-spec underground_system_check_out(Id :: integer(), StationName ::
unicode:unicode_binary(), T :: integer()) -> any().
underground_system_check_out(Id, StationName, T) ->
.

-spec underground_system_get_average_time(StartStation ::
unicode:unicode_binary(), EndStation :: unicode:unicode_binary()) -> float().
underground_system_get_average_time(StartStation, EndStation) ->
.
```

```
%% Your functions will be called as such:
%% underground_system_init_(),
%% underground_system_check_in(Id, StationName, T),
%% underground_system_check_out(Id, StationName, T),
%% Param_3 = underground_system_get_average_time(StartStation, EndStation),

%% underground_system_init_ will be called before every test case, in which
you can do some necessary initializations.
```

Racket:

```
(define underground-system%
(class object%
(super-new)

(init-field)

; check-in : exact-integer? string? exact-integer? -> void?
(define/public (check-in id station-name t)
)
; check-out : exact-integer? string? exact-integer? -> void?
(define/public (check-out id station-name t)
)
; get-average-time : string? string? -> flonum?
(define/public (get-average-time start-station end-station)
)))

;; Your underground-system% object will be instantiated and called as such:
;; (define obj (new underground-system%))
;; (send obj check-in id station-name t)
;; (send obj check-out id station-name t)
;; (define param_3 (send obj get-average-time start-station end-station))
```

Solutions

C++ Solution:

```
/*
* Problem: Design Underground System
```

```
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class UndergroundSystem {
public:
UndergroundSystem() {

}

void checkIn(int id, string stationName, int t) {

}

void checkOut(int id, string stationName, int t) {

}

double getAverageTime(string startStation, string endStation) {

}
};

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem* obj = new UndergroundSystem();
* obj->checkIn(id,stationName,t);
* obj->checkOut(id,stationName,t);
* double param_3 = obj->getAverageTime(startStation,endStation);
*/
```

Java Solution:

```
/**
* Problem: Design Underground System
* Difficulty: Medium
* Tags: string, hash
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class UndergroundSystem {

public UndergroundSystem() {

}

public void checkIn(int id, String stationName, int t) {

}

public void checkOut(int id, String stationName, int t) {

}

public double getAverageTime(String startStation, String endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem obj = new UndergroundSystem();
* obj.checkIn(id,stationName,t);
* obj.checkOut(id,stationName,t);
* double param_3 = obj.getAverageTime(startStation,endStation);
*/
```

Python3 Solution:

```
"""
Problem: Design Underground System
Difficulty: Medium
Tags: string, hash

Approach: String manipulation with hash map or two pointers
```

```
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class UndergroundSystem:

def __init__(self):


def checkIn(self, id: int, stationName: str, t: int) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class UndergroundSystem(object):

def __init__(self):


def checkIn(self, id, stationName, t):
"""
:type id: int
:type stationName: str
:type t: int
:rtype: None
"""


def checkOut(self, id, stationName, t):
"""
:type id: int
:type stationName: str
:type t: int
:rtype: None
"""


def getAverageTime(self, startStation, endStation):
"""
:type startStation: str
```

```
:type endStation: str
:rtype: float
"""



# Your UndergroundSystem object will be instantiated and called as such:
# obj = UndergroundSystem()
# obj.checkIn(id,stationName,t)
# obj.checkOut(id,stationName,t)
# param_3 = obj.getAverageTime(startStation,endStation)
```

JavaScript Solution:

```
/**
* Problem: Design Underground System
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/


var UndergroundSystem = function() {

};

/**
* @param {number} id
* @param {string} stationName
* @param {number} t
* @return {void}
*/
UndergroundSystem.prototype.checkIn = function(id, stationName, t) {

};

/**
* @param {number} id
```

```
* @param {string} stationName
* @param {number} t
* @return {void}
*/
UndergroundSystem.prototype.checkOut = function(id, stationName, t) {

};

/**
* @param {string} startStation
* @param {string} endStation
* @return {number}
*/
UndergroundSystem.prototype.getAverageTime = function(startStation,
endStation) {

};

/**
* Your UndergroundSystem object will be instantiated and called as such:
* var obj = new UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* var param_3 = obj.getAverageTime(startStation,endStation)
*/
```

TypeScript Solution:

```
/**
* Problem: Design Underground System
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class UndergroundSystem {
constructor() {
```

```
}

checkIn(id: number, stationName: string, t: number): void {

}

checkOut(id: number, stationName: string, t: number): void {

}

getAverageTime(startStation: string, endStation: string): number {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* var obj = new UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* var param_3 = obj.getAverageTime(startStation,endStation)
*/
```

C# Solution:

```
/*
* Problem: Design Underground System
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class UndergroundSystem {

public UndergroundSystem() {

}
```

```
public void CheckIn(int id, string stationName, int t) {

}

public void CheckOut(int id, string stationName, int t) {

}

public double GetAverageTime(string startStation, string endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem obj = new UndergroundSystem();
* obj.CheckIn(id,stationName,t);
* obj.CheckOut(id,stationName,t);
* double param_3 = obj.GetAverageTime(startStation,endStation);
*/
```

C Solution:

```
/*
* Problem: Design Underground System
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/




typedef struct {

} UndergroundSystem;
```

```
UndergroundSystem* undergroundSystemCreate() {

}

void undergroundSystemCheckIn(UndergroundSystem* obj, int id, char*
stationName, int t) {

}

void undergroundSystemCheckOut(UndergroundSystem* obj, int id, char*
stationName, int t) {

}

double undergroundSystemGetAverageTime(UndergroundSystem* obj, char*
startStation, char* endStation) {

}

void undergroundSystemFree(UndergroundSystem* obj) {

}

/**
* Your UndergroundSystem struct will be instantiated and called as such:
* UndergroundSystem* obj = undergroundSystemCreate();
* undergroundSystemCheckIn(obj, id, stationName, t);

* undergroundSystemCheckOut(obj, id, stationName, t);

* double param_3 = undergroundSystemGetAverageTime(obj, startStation,
endStation);

* undergroundSystemFree(obj);
*/
```

Go Solution:

```
// Problem: Design Underground System
// Difficulty: Medium
// Tags: string, hash
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

type UndergroundSystem struct {

}


func Constructor() UndergroundSystem {

}


func (this *UndergroundSystem) CheckIn(id int, stationName string, t int) {

}


func (this *UndergroundSystem) CheckOut(id int, stationName string, t int) {

}


func (this *UndergroundSystem) GetAverageTime(startStation string, endStation
string) float64 {

}


/**
* Your UndergroundSystem object will be instantiated and called as such:
* obj := Constructor();
* obj.CheckIn(id,stationName,t);
* obj.CheckOut(id,stationName,t);
* param_3 := obj.GetAverageTime(startStation,endStation);
*/
```

Kotlin Solution:

```
class UndergroundSystem() {

fun checkIn(id: Int, stationName: String, t: Int) {

}

fun checkOut(id: Int, stationName: String, t: Int) {

}

fun getAverageTime(startStation: String, endStation: String): Double {

}

}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* var obj = UndergroundSystem()
* obj.checkIn(id,stationName,t)
* obj.checkOut(id,stationName,t)
* var param_3 = obj.getAverageTime(startStation,endStation)
*/
```

Swift Solution:

```
class UndergroundSystem {

init() {

}

func checkIn(_ id: Int, _ stationName: String, _ t: Int) {

}

func checkOut(_ id: Int, _ stationName: String, _ t: Int) {

}

func getAverageTime(_ startStation: String, _ endStation: String) -> Double {
```

```
}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* let obj = UndergroundSystem()
* obj.checkIn(id, stationName, t)
* obj.checkOut(id, stationName, t)
* let ret_3: Double = obj.getAverageTime(startStation, endStation)
*/
```

Rust Solution:

```
// Problem: Design Underground System
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

struct UndergroundSystem {

}



/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl UndergroundSystem {

fn new() -> Self {

}

fn check_in(&self, id: i32, station_name: String, t: i32) {

}
```

```
fn check_out(&self, id: i32, station_name: String, t: i32) {

}

fn get_average_time(&self, start_station: String, end_station: String) -> f64
{

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* let obj = UndergroundSystem::new();
* obj.check_in(id, stationName, t);
* obj.check_out(id, stationName, t);
* let ret_3: f64 = obj.get_average_time(startStation, endStation);
*/
```

Ruby Solution:

```
class UndergroundSystem
def initialize()

end


=begin
:type id: Integer
:type station_name: String
:type t: Integer
:rtype: Void
=end
def check_in(id, station_name, t)

end


=begin
:type id: Integer
:type station_name: String
:type t: Integer
```

```
:rtype: Void
=end
def check_out(id, station_name, t)

end


=begin
:type start_station: String
:type end_station: String
:rtype: Float
=end
def get_average_time(start_station, end_station)

end



end

# Your UndergroundSystem object will be instantiated and called as such:
# obj = UndergroundSystem.new()
# obj.check_in(id, station_name, t)
# obj.check_out(id, station_name, t)
# param_3 = obj.get_average_time(start_station, end_station)
```

PHP Solution:

```
class UndergroundSystem {
/**
*/
function __construct() {

}

/**
* @param Integer $id
* @param String $stationName
* @param Integer $t
* @return NULL
*/
function checkIn($id, $stationName, $t) {
```

```
}

/**
* @param Integer $id
* @param String $stationName
* @param Integer $t
* @return NULL
*/
function checkOut($id, $stationName, $t) {

}

/**
* @param String $startStation
* @param String $endStation
* @return Float
*/
function getAverageTime($startStation, $endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* $obj = UndergroundSystem();
* $obj->checkIn($id, $stationName, $t);
* $obj->checkOut($id, $stationName, $t);
* $ret_3 = $obj->getAverageTime($startStation, $endStation);
*/
```

Dart Solution:

```
class UndergroundSystem {

UndergroundSystem() {

}

void checkIn(int id, String stationName, int t) {
```

```
}

void checkOut(int id, String stationName, int t) {

}

double getAverageTime(String startStation, String endStation) {

}
}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* UndergroundSystem obj = UndergroundSystem();
* obj.checkIn(id,stationName,t);
* obj.checkOut(id,stationName,t);
* double param3 = obj.getAverageTime(startStation,endStation);
*/
```

Scala Solution:

```
class UndergroundSystem() {

def checkIn(id: Int, stationName: String, t: Int): Unit = {

}

def checkOut(id: Int, stationName: String, t: Int): Unit = {

}

def getAverageTime(startStation: String, endStation: String): Double = {

}

}

/**
* Your UndergroundSystem object will be instantiated and called as such:
* val obj = new UndergroundSystem()
* obj.checkIn(id,stationName,t)
```

```
* obj.checkOut(id,stationName,t)
* val param_3 = obj.getAverageTime(startStation,endStation)
*/
```

Elixir Solution:

```
defmodule UndergroundSystem do
@spec init_() :: any
def init_() do

end

@spec check_in(id :: integer, station_name :: String.t, t :: integer) :: any
def check_in(id, station_name, t) do

end

@spec check_out(id :: integer, station_name :: String.t, t :: integer) :: any
def check_out(id, station_name, t) do

end

@spec get_average_time(start_station :: String.t, end_station :: String.t) ::
float
def get_average_time(start_station, end_station) do

end
end

# Your functions will be called as such:
# UndergroundSystem.init_()
# UndergroundSystem.check_in(id, station_name, t)
# UndergroundSystem.check_out(id, station_name, t)
# param_3 = UndergroundSystem.get_average_time(start_station, end_station)

# UndergroundSystem.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang Solution:

```
-spec underground_system_init_() -> any().
underground_system_init_() ->
.

-spec underground_system_check_in(Id :: integer(), StationName ::
unicode:unicode_binary(), T :: integer()) -> any().
underground_system_check_in(Id, StationName, T) ->
.

-spec underground_system_check_out(Id :: integer(), StationName ::
unicode:unicode_binary(), T :: integer()) -> any().
underground_system_check_out(Id, StationName, T) ->
.

-spec underground_system_get_average_time(StartStation ::
unicode:unicode_binary(), EndStation :: unicode:unicode_binary()) -> float().
underground_system_get_average_time(StartStation, EndStation) ->
.


%% Your functions will be called as such:
%% underground_system_init_(),
%% underground_system_check_in(Id, StationName, T),
%% underground_system_check_out(Id, StationName, T),
%% Param_3 = underground_system_get_average_time(StartStation, EndStation),

%% underground_system_init_ will be called before every test case, in which
you can do some necessary initializations.
```

Racket Solution:

```
(define underground-system%
(class object%
(super-new)

(init-field)

; check-in : exact-integer? string? exact-integer? -> void?
(define/public (check-in id station-name t)
)
; check-out : exact-integer? string? exact-integer? -> void?
(define/public (check-out id station-name t)
```

```
)
; get-average-time : string? string? -> flonum?
(define/public (get-average-time start-station end-station)
)))

;; Your underground-system% object will be instantiated and called as such:
;; (define obj (new underground-system%))
;; (send obj check-in id station-name t)
;; (send obj check-out id station-name t)
;; (define param_3 (send obj get-average-time start-station end-station))
```