Problem 3369: Design an Array Statistics Tracker

Problem Information

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

Problem Description

Design a data structure that keeps track of the values in it and answers some queries regarding their mean, median, and mode.

Implement the

StatisticsTracker

class.

StatisticsTracker()

: Initialize the

StatisticsTracker

object with an empty array.

void addNumber(int number)

: Add

number

to the data structure.

void removeFirstAddedNumber()

: Remove the earliest added number from the data structure.

int getMean()

: Return the floored

mean

of the numbers in the data structure.

int getMedian()

: Return the

median

of the numbers in the data structure.

int getMode()

: Return the

mode

of the numbers in the data structure. If there are multiple modes, return the smallest one.

Note

:

The

mean

of an array is the sum of all the values divided by the number of values in the array.

The

median

of an array is the middle element of the array when it is sorted in non-decreasing order. If there are two choices for a median, the larger of the two values is taken.

The

mode

of an array is the element that appears most often in the array.

Example 1:

Input:

["StatisticsTracker", "addNumber", "addNumber", "addNumber", "addNumber", "getMean", "getMedian", "getMode", "removeFirstAddedNumber", "getMode"]

[[], [4], [4], [2], [3], [], [], [], [], []]

Output:

[null, null, null, null, null, 3, 4, 4, null, 2]

Explanation

StatisticsTracker statisticsTracker = new StatisticsTracker();

statisticsTracker.addNumber(4); // The data structure now contains [4]

statisticsTracker.addNumber(4); // The data structure now contains [4, 4]

statisticsTracker.addNumber(2); // The data structure now contains [4, 4, 2]

statisticsTracker.addNumber(3); // The data structure now contains [4, 4, 2, 3]

statisticsTracker.getMean(); // return 3

statisticsTracker.getMedian(); // return 4

statisticsTracker.getMode(); // return 4

statisticsTracker.removeFirstAddedNumber(); // The data structure now contains [4, 2, 3]

statisticsTracker.getMode(); // return 2

Example 2:

Input:

["StatisticsTracker", "addNumber", "addNumber", "getMean", "removeFirstAddedNumber", "addNumber", "addNumber", "removeFirstAddedNumber", "getMedian", "addNumber", "getMode"]

[[], [9], [5], [], [], [5], [6], [], [], [8], []]

Output:

[null, null, null, 7, null, null, null, null, 6, null, 5]

Explanation

StatisticsTracker statisticsTracker = new StatisticsTracker();

statisticsTracker.addNumber(9); // The data structure now contains [9]

statisticsTracker.addNumber(5); // The data structure now contains [9, 5]

statisticsTracker.getMean(); // return 7

statisticsTracker.removeFirstAddedNumber(); // The data structure now contains [5]

statisticsTracker.addNumber(5); // The data structure now contains [5, 5]

statisticsTracker.addNumber(6); // The data structure now contains [5, 5, 6]

statisticsTracker.removeFirstAddedNumber(); // The data structure now contains [5, 6]

statisticsTracker.getMedian(); // return 6

statisticsTracker.addNumber(8); // The data structure now contains [5, 6, 8]

statisticsTracker.getMode(); // return 5

Constraints:

1 <= number <= 10

9

At most,

10

5

calls will be made to

addNumber

,

removeFirstAddedNumber

,

getMean

,

getMedian

, and

getMode

in total.

removeFirstAddedNumber

,

getMean

,

getMedian

, and

getMode

will be called only if there is at least one element in the data structure.

Code Snippets

C++:

```
class StatisticsTracker {
public:
StatisticsTracker() {

}

void addNumber(int number) {

}

void removeFirstAddedNumber() {

}

int getMean() {

}

int getMedian() {
```

```
}

int getMode() {

}
};

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker* obj = new StatisticsTracker();
* obj->addNumber(number);
* obj->removeFirstAddedNumber();
* int param_3 = obj->getMean();
* int param_4 = obj->getMedian();
* int param_5 = obj->getMode();
*/
```

Java:

```
class StatisticsTracker {

public StatisticsTracker() {

}

public void addNumber(int number) {

}

public void removeFirstAddedNumber() {

}

public int getMean() {

}

public int getMedian() {

}

public int getMode() {
```

```
}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker obj = new StatisticsTracker();
* obj.addNumber(number);
* obj.removeFirstAddedNumber();
* int param_3 = obj.getMean();
* int param_4 = obj.getMedian();
* int param_5 = obj.getMode();
*/
```

Python3:

```
class StatisticsTracker:

def __init__(self):


def addNumber(self, number: int) -> None:


def removeFirstAddedNumber(self) -> None:


def getMean(self) -> int:


def getMedian(self) -> int:


def getMode(self) -> int:



# Your StatisticsTracker object will be instantiated and called as such:
# obj = StatisticsTracker()
# obj.addNumber(number)
# obj.removeFirstAddedNumber()
# param_3 = obj.getMean()
```

```
# param_4 = obj.getMedian()
# param_5 = obj.getMode()
```

Python:

```
class StatisticsTracker(object):

def __init__(self):


def addNumber(self, number):
"""
:type number: int
:rtype: None
"""


def removeFirstAddedNumber(self):
"""
:rtype: None
"""


def getMean(self):
"""
:rtype: int
"""


def getMedian(self):
"""
:rtype: int
"""


def getMode(self):
"""
:rtype: int
"""
```

```
# Your StatisticsTracker object will be instantiated and called as such:
# obj = StatisticsTracker()
# obj.addNumber(number)
# obj.removeFirstAddedNumber()
# param_3 = obj.getMean()
# param_4 = obj.getMedian()
# param_5 = obj.getMode()
```

JavaScript:

```
var StatisticsTracker = function() {

};

/**
* @param {number} number
* @return {void}
*/
StatisticsTracker.prototype.addNumber = function(number) {

};

/**
* @return {void}
*/
StatisticsTracker.prototype.removeFirstAddedNumber = function() {

};

/**
* @return {number}
*/
StatisticsTracker.prototype.getMean = function() {

};

/**
* @return {number}
*/
StatisticsTracker.prototype.getMedian = function() {
```

```
};

/**
* @return {number}
*/
StatisticsTracker.prototype.getMode = function() {

};

/**
* Your StatisticsTracker object will be instantiated and called as such:
* var obj = new StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* var param_3 = obj.getMean()
* var param_4 = obj.getMedian()
* var param_5 = obj.getMode()
*/
```

TypeScript:

```
class StatisticsTracker {
constructor() {

}

addNumber(number: number): void {

}

removeFirstAddedNumber(): void {

}

getMean(): number {

}

getMedian(): number {

}
```

```
getMode(): number {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* var obj = new StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* var param_3 = obj.getMean()
* var param_4 = obj.getMedian()
* var param_5 = obj.getMode()
*/
```

C#:

```
public class StatisticsTracker {

public StatisticsTracker() {

}

public void AddNumber(int number) {

}

public void RemoveFirstAddedNumber() {

}

public int GetMean() {

}

public int GetMedian() {

}

public int GetMode() {

}
```

```
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker obj = new StatisticsTracker();
* obj.AddNumber(number);
* obj.RemoveFirstAddedNumber();
* int param_3 = obj.GetMean();
* int param_4 = obj.GetMedian();
* int param_5 = obj.GetMode();
*/
```

C:

```
typedef struct {

} StatisticsTracker;


StatisticsTracker* statisticsTrackerCreate() {

}

void statisticsTrackerAddNumber(StatisticsTracker* obj, int number) {

}

void statisticsTrackerRemoveFirstAddedNumber(StatisticsTracker* obj) {

}

int statisticsTrackerGetMean(StatisticsTracker* obj) {

}

int statisticsTrackerGetMedian(StatisticsTracker* obj) {

}
```

```
int statisticsTrackerGetMode(StatisticsTracker* obj) {

}

void statisticsTrackerFree(StatisticsTracker* obj) {

}

/**
* Your StatisticsTracker struct will be instantiated and called as such:
* StatisticsTracker* obj = statisticsTrackerCreate();
* statisticsTrackerAddNumber(obj, number);

* statisticsTrackerRemoveFirstAddedNumber(obj);

* int param_3 = statisticsTrackerGetMean(obj);

* int param_4 = statisticsTrackerGetMedian(obj);

* int param_5 = statisticsTrackerGetMode(obj);

* statisticsTrackerFree(obj);
*/
```

Go:

```
type StatisticsTracker struct {

}


func Constructor() StatisticsTracker {

}


func (this *StatisticsTracker) AddNumber(number int) {

}


func (this *StatisticsTracker) RemoveFirstAddedNumber() {
```

```
}


func (this *StatisticsTracker) GetMean() int {

}


func (this *StatisticsTracker) GetMedian() int {

}


func (this *StatisticsTracker) GetMode() int {

}


/**
* Your StatisticsTracker object will be instantiated and called as such:
* obj := Constructor();
* obj.AddNumber(number);
* obj.RemoveFirstAddedNumber();
* param_3 := obj.GetMean();
* param_4 := obj.GetMedian();
* param_5 := obj.GetMode();
*/
```

Kotlin:

```
class StatisticsTracker() {

fun addNumber(number: Int) {

}

fun removeFirstAddedNumber() {

}

fun getMean(): Int {
```

```
}

fun getMedian(): Int {

}

fun getMode(): Int {

}

}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* var obj = StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* var param_3 = obj.getMean()
* var param_4 = obj.getMedian()
* var param_5 = obj.getMode()
*/
```

Swift:

```
class StatisticsTracker {

init() {

}

func addNumber(_ number: Int) {

}

func removeFirstAddedNumber() {

}

func getMean() -> Int {
```

```
}

func getMedian() -> Int {

}

func getMode() -> Int {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* let obj = StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* let ret_3: Int = obj.getMean()
* let ret_4: Int = obj.getMedian()
* let ret_5: Int = obj.getMode()
*/
```

Rust:

```
struct StatisticsTracker {

}


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

fn new() -> Self {

}

fn add_number(&self, number: i32) {

}
```

```
fn remove_first_added_number(&self) {

}

fn get_mean(&self) -> i32 {

}

fn get_median(&self) -> i32 {

}

fn get_mode(&self) -> i32 {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* let obj = StatisticsTracker::new();
* obj.add_number(number);
* obj.remove_first_added_number();
* let ret_3: i32 = obj.get_mean();
* let ret_4: i32 = obj.get_median();
* let ret_5: i32 = obj.get_mode();
*/
```

Ruby:

```
class StatisticsTracker
def initialize()

end


=begin
:type number: Integer
:rtype: Void
=end
def add_number(number)

end
```

```
=begin
:rtype: Void
=end
def remove_first_added_number()

end


=begin
:rtype: Integer
=end
def get_mean()

end


=begin
:rtype: Integer
=end
def get_median()

end


=begin
:rtype: Integer
=end
def get_mode()

end


end

# Your StatisticsTracker object will be instantiated and called as such:
# obj = StatisticsTracker.new()
# obj.add_number(number)
# obj.remove_first_added_number()
# param_3 = obj.get_mean()
# param_4 = obj.get_median()
```

```
# param_5 = obj.get_mode()
```

PHP:

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

}

/**
* @param Integer $number
* @return NULL
*/
function addNumber($number) {

}

/**
* @return NULL
*/
function removeFirstAddedNumber() {

}

/**
* @return Integer
*/
function getMean() {

}

/**
* @return Integer
*/
function getMedian() {

}

/**
* @return Integer
```

```
*/
function getMode() {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* $obj = StatisticsTracker();
* $obj->addNumber($number);
* $obj->removeFirstAddedNumber();
* $ret_3 = $obj->getMean();
* $ret_4 = $obj->getMedian();
* $ret_5 = $obj->getMode();
*/
```

Dart:

```
class StatisticsTracker {

StatisticsTracker() {

}

void addNumber(int number) {

}

void removeFirstAddedNumber() {

}

int getMean() {

}

int getMedian() {

}

int getMode() {
```

```
}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker obj = StatisticsTracker();
* obj.addNumber(number);
* obj.removeFirstAddedNumber();
* int param3 = obj.getMean();
* int param4 = obj.getMedian();
* int param5 = obj.getMode();
*/
```

Scala:

```
class StatisticsTracker() {

def addNumber(number: Int): Unit = {

}

def removeFirstAddedNumber(): Unit = {

}

def getMean(): Int = {

}

def getMedian(): Int = {

}

def getMode(): Int = {

}

}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* val obj = new StatisticsTracker()
```

```
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* val param_3 = obj.getMean()
* val param_4 = obj.getMedian()
* val param_5 = obj.getMode()
*/
```

Elixir:

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

end

@spec add_number(number :: integer) :: any
def add_number(number) do

end

@spec remove_first_added_number() :: any
def remove_first_added_number() do

end

@spec get_mean() :: integer
def get_mean() do

end

@spec get_median() :: integer
def get_median() do

end

@spec get_mode() :: integer
def get_mode() do

end
end

# Your functions will be called as such:
```

```
# StatisticsTracker.init_()
# StatisticsTracker.add_number(number)
# StatisticsTracker.remove_first_added_number()
# param_3 = StatisticsTracker.get_mean()
# param_4 = StatisticsTracker.get_median()
# param_5 = StatisticsTracker.get_mode()

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

Erlang:

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

-spec statistics_tracker_add_number(Number :: integer()) -> any().
statistics_tracker_add_number(Number) ->
.

-spec statistics_tracker_remove_first_added_number() -> any().
statistics_tracker_remove_first_added_number() ->
.

-spec statistics_tracker_get_mean() -> integer().
statistics_tracker_get_mean() ->
.

-spec statistics_tracker_get_median() -> integer().
statistics_tracker_get_median() ->
.

-spec statistics_tracker_get_mode() -> integer().
statistics_tracker_get_mode() ->
.


%% Your functions will be called as such:
%% statistics_tracker_init_(),
%% statistics_tracker_add_number(Number),
%% statistics_tracker_remove_first_added_number(),
%% Param_3 = statistics_tracker_get_mean(),
```

```
%% Param_4 = statistics_tracker_get_median(),
%% Param_5 = statistics_tracker_get_mode(),

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

Racket:

```
(define statistics-tracker%
(class object%
(super-new)

(init-field)

; add-number : exact-integer? -> void?
(define/public (add-number number)
)
; remove-first-added-number : -> void?
(define/public (remove-first-added-number)
)
; get-mean : -> exact-integer?
(define/public (get-mean)
)
; get-median : -> exact-integer?
(define/public (get-median)
)
; get-mode : -> exact-integer?
(define/public (get-mode)
)))

;; Your statistics-tracker% object will be instantiated and called as such:
;; (define obj (new statistics-tracker%))
;; (send obj add-number number)
;; (send obj remove-first-added-number)
;; (define param_3 (send obj get-mean))
;; (define param_4 (send obj get-median))
;; (define param_5 (send obj get-mode))
```

Solutions

C++ Solution:

```
/*
* Problem: Design an Array Statistics Tracker
* Difficulty: Hard
* Tags: array, hash, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class StatisticsTracker {
public:
StatisticsTracker() {

}

void addNumber(int number) {

}

void removeFirstAddedNumber() {

}

int getMean() {

}

int getMedian() {

}

int getMode() {

}
};

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker* obj = new StatisticsTracker();
* obj->addNumber(number);
```

```
* obj->removeFirstAddedNumber();
* int param_3 = obj->getMean();
* int param_4 = obj->getMedian();
* int param_5 = obj->getMode();
*/
```

Java Solution:

```
/**
* Problem: Design an Array Statistics Tracker
* Difficulty: Hard
* Tags: array, hash, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class StatisticsTracker {

public StatisticsTracker() {

}

public void addNumber(int number) {

}

public void removeFirstAddedNumber() {

}

public int getMean() {

}

public int getMedian() {

}

public int getMode() {
```

```
}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker obj = new StatisticsTracker();
* obj.addNumber(number);
* obj.removeFirstAddedNumber();
* int param_3 = obj.getMean();
* int param_4 = obj.getMedian();
* int param_5 = obj.getMode();
*/
```

Python3 Solution:

```
"""
Problem: Design an Array Statistics Tracker
Difficulty: Hard
Tags: array, hash, sort, search, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class StatisticsTracker:

def __init__(self):


def addNumber(self, number: int) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class StatisticsTracker(object):

def __init__(self):
```

```
def addNumber(self, number):
"""
:type number: int
:rtype: None
"""


def removeFirstAddedNumber(self):
"""
:rtype: None
"""


def getMean(self):
"""
:rtype: int
"""


def getMedian(self):
"""
:rtype: int
"""


def getMode(self):
"""
:rtype: int
"""



# Your StatisticsTracker object will be instantiated and called as such:
# obj = StatisticsTracker()
# obj.addNumber(number)
# obj.removeFirstAddedNumber()
# param_3 = obj.getMean()
# param_4 = obj.getMedian()
# param_5 = obj.getMode()
```

JavaScript Solution:

```
/**
* Problem: Design an Array Statistics Tracker
* Difficulty: Hard
* Tags: array, hash, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/


var StatisticsTracker = function() {

};

/**
* @param {number} number
* @return {void}
*/
StatisticsTracker.prototype.addNumber = function(number) {

};

/**
* @return {void}
*/
StatisticsTracker.prototype.removeFirstAddedNumber = function() {

};

/**
* @return {number}
*/
StatisticsTracker.prototype.getMean = function() {

};

/**
* @return {number}
*/
StatisticsTracker.prototype.getMedian = function() {
```

```
};

/**
* @return {number}
*/
StatisticsTracker.prototype.getMode = function() {

};

/**
* Your StatisticsTracker object will be instantiated and called as such:
* var obj = new StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* var param_3 = obj.getMean()
* var param_4 = obj.getMedian()
* var param_5 = obj.getMode()
*/
```

TypeScript Solution:

```
/**
* Problem: Design an Array Statistics Tracker
* Difficulty: Hard
* Tags: array, hash, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class StatisticsTracker {
constructor() {

}

addNumber(number: number): void {

}
```

```
removeFirstAddedNumber(): void {

}

getMean(): number {

}

getMedian(): number {

}

getMode(): number {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* var obj = new StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* var param_3 = obj.getMean()
* var param_4 = obj.getMedian()
* var param_5 = obj.getMode()
*/
```

C# Solution:

```
/*
* Problem: Design an Array Statistics Tracker
* Difficulty: Hard
* Tags: array, hash, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class StatisticsTracker {
```

```
public StatisticsTracker() {

}

public void AddNumber(int number) {

}

public void RemoveFirstAddedNumber() {

}

public int GetMean() {

}

public int GetMedian() {

}

public int GetMode() {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker obj = new StatisticsTracker();
* obj.AddNumber(number);
* obj.RemoveFirstAddedNumber();
* int param_3 = obj.GetMean();
* int param_4 = obj.GetMedian();
* int param_5 = obj.GetMode();
*/
```

C Solution:

```
/*
* Problem: Design an Array Statistics Tracker
* Difficulty: Hard
* Tags: array, hash, sort, search, queue, heap
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/



typedef struct {

} StatisticsTracker;


StatisticsTracker* statisticsTrackerCreate() {

}

void statisticsTrackerAddNumber(StatisticsTracker* obj, int number) {

}

void statisticsTrackerRemoveFirstAddedNumber(StatisticsTracker* obj) {

}

int statisticsTrackerGetMean(StatisticsTracker* obj) {

}

int statisticsTrackerGetMedian(StatisticsTracker* obj) {

}

int statisticsTrackerGetMode(StatisticsTracker* obj) {

}

void statisticsTrackerFree(StatisticsTracker* obj) {

}
```

```
/**
* Your StatisticsTracker struct will be instantiated and called as such:
* StatisticsTracker* obj = statisticsTrackerCreate();
* statisticsTrackerAddNumber(obj, number);

* statisticsTrackerRemoveFirstAddedNumber(obj);

* int param_3 = statisticsTrackerGetMean(obj);

* int param_4 = statisticsTrackerGetMedian(obj);

* int param_5 = statisticsTrackerGetMode(obj);

* statisticsTrackerFree(obj);
*/
```

Go Solution:

```
// Problem: Design an Array Statistics Tracker
// Difficulty: Hard
// Tags: array, hash, sort, search, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

type StatisticsTracker struct {

}


func Constructor() StatisticsTracker {

}


func (this *StatisticsTracker) AddNumber(number int) {

}
```

```
func (this *StatisticsTracker) RemoveFirstAddedNumber() {

}


func (this *StatisticsTracker) GetMean() int {

}


func (this *StatisticsTracker) GetMedian() int {

}


func (this *StatisticsTracker) GetMode() int {

}


/**
* Your StatisticsTracker object will be instantiated and called as such:
* obj := Constructor();
* obj.AddNumber(number);
* obj.RemoveFirstAddedNumber();
* param_3 := obj.GetMean();
* param_4 := obj.GetMedian();
* param_5 := obj.GetMode();
*/
```

Kotlin Solution:

```
class StatisticsTracker() {

fun addNumber(number: Int) {

}

fun removeFirstAddedNumber() {

}
```

```
fun getMean(): Int {

}

fun getMedian(): Int {

}

fun getMode(): Int {

}

}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* var obj = StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* var param_3 = obj.getMean()
* var param_4 = obj.getMedian()
* var param_5 = obj.getMode()
*/
```

Swift Solution:

```
class StatisticsTracker {

init() {

}

func addNumber(_ number: Int) {

}

func removeFirstAddedNumber() {

}
```

```
func getMean() -> Int {

}

func getMedian() -> Int {

}

func getMode() -> Int {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* let obj = StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* let ret_3: Int = obj.getMean()
* let ret_4: Int = obj.getMedian()
* let ret_5: Int = obj.getMode()
*/
```

Rust Solution:

```
// Problem: Design an Array Statistics Tracker
// Difficulty: Hard
// Tags: array, hash, sort, search, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

struct StatisticsTracker {

}



/**
* `&self` means the method takes an immutable reference.
```

```
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl StatisticsTracker {

fn new() -> Self {

}

fn add_number(&self, number: i32) {

}

fn remove_first_added_number(&self) {

}

fn get_mean(&self) -> i32 {

}

fn get_median(&self) -> i32 {

}

fn get_mode(&self) -> i32 {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* let obj = StatisticsTracker::new();
* obj.add_number(number);
* obj.remove_first_added_number();
* let ret_3: i32 = obj.get_mean();
* let ret_4: i32 = obj.get_median();
* let ret_5: i32 = obj.get_mode();
*/
```

Ruby Solution:

```
class StatisticsTracker
def initialize()

end


=begin
:type number: Integer
:rtype: Void
=end
def add_number(number)

end


=begin
:rtype: Void
=end
def remove_first_added_number()

end


=begin
:rtype: Integer
=end
def get_mean()

end


=begin
:rtype: Integer
=end
def get_median()

end


=begin
:rtype: Integer
=end
def get_mode()
```

```
end


end

# Your StatisticsTracker object will be instantiated and called as such:
# obj = StatisticsTracker.new()
# obj.add_number(number)
# obj.remove_first_added_number()
# param_3 = obj.get_mean()
# param_4 = obj.get_median()
# param_5 = obj.get_mode()
```

PHP Solution:

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

}

/**
* @param Integer $number
* @return NULL
*/
function addNumber($number) {

}

/**
* @return NULL
*/
function removeFirstAddedNumber() {

}

/**
* @return Integer
*/
```

```
function getMean() {

}

/**
* @return Integer
*/
function getMedian() {

}

/**
* @return Integer
*/
function getMode() {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* $obj = StatisticsTracker();
* $obj->addNumber($number);
* $obj->removeFirstAddedNumber();
* $ret_3 = $obj->getMean();
* $ret_4 = $obj->getMedian();
* $ret_5 = $obj->getMode();
*/
```

Dart Solution:

```
class StatisticsTracker {

StatisticsTracker() {

}

void addNumber(int number) {

}
```

```
void removeFirstAddedNumber() {

}

int getMean() {

}

int getMedian() {

}

int getMode() {

}
}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* StatisticsTracker obj = StatisticsTracker();
* obj.addNumber(number);
* obj.removeFirstAddedNumber();
* int param3 = obj.getMean();
* int param4 = obj.getMedian();
* int param5 = obj.getMode();
*/
```

Scala Solution:

```
class StatisticsTracker() {

def addNumber(number: Int): Unit = {

}

def removeFirstAddedNumber(): Unit = {

}

def getMean(): Int = {
```

```
}

def getMedian(): Int = {

}

def getMode(): Int = {

}

}

/**
* Your StatisticsTracker object will be instantiated and called as such:
* val obj = new StatisticsTracker()
* obj.addNumber(number)
* obj.removeFirstAddedNumber()
* val param_3 = obj.getMean()
* val param_4 = obj.getMedian()
* val param_5 = obj.getMode()
*/
```

Elixir Solution:

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

end

@spec add_number(number :: integer) :: any
def add_number(number) do

end

@spec remove_first_added_number() :: any
def remove_first_added_number() do

end

@spec get_mean() :: integer
```

```
def get_mean() do

end

@spec get_median() :: integer
def get_median() do

end

@spec get_mode() :: integer
def get_mode() do

end
end

# Your functions will be called as such:
# StatisticsTracker.init_()
# StatisticsTracker.add_number(number)
# StatisticsTracker.remove_first_added_number()
# param_3 = StatisticsTracker.get_mean()
# param_4 = StatisticsTracker.get_median()
# param_5 = StatisticsTracker.get_mode()

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

Erlang Solution:

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

-spec statistics_tracker_add_number(Number :: integer()) -> any().
statistics_tracker_add_number(Number) ->
.

-spec statistics_tracker_remove_first_added_number() -> any().
statistics_tracker_remove_first_added_number() ->
.

-spec statistics_tracker_get_mean() -> integer().
```

```
statistics_tracker_get_mean() ->
.

-spec statistics_tracker_get_median() -> integer().
statistics_tracker_get_median() ->
.

-spec statistics_tracker_get_mode() -> integer().
statistics_tracker_get_mode() ->
.


%% Your functions will be called as such:
%% statistics_tracker_init_(),
%% statistics_tracker_add_number(Number),
%% statistics_tracker_remove_first_added_number(),
%% Param_3 = statistics_tracker_get_mean(),
%% Param_4 = statistics_tracker_get_median(),
%% Param_5 = statistics_tracker_get_mode(),

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

Racket Solution:

```
(define statistics-tracker%
(class object%
(super-new)

(init-field)

; add-number : exact-integer? -> void?
(define/public (add-number number)
)
; remove-first-added-number : -> void?
(define/public (remove-first-added-number)
)
; get-mean : -> exact-integer?
(define/public (get-mean)
)
; get-median : -> exact-integer?
```

```
(define/public (get-median)
)
; get-mode : -> exact-integer?
(define/public (get-mode)
)))

;; Your statistics-tracker% object will be instantiated and called as such:
;; (define obj (new statistics-tracker%))
;; (send obj add-number number)
;; (send obj remove-first-added-number)
;; (define param_3 (send obj get-mean))
;; (define param_4 (send obj get-median))
;; (define param_5 (send obj get-mode))
```