Problem 1114: Print in Order

Problem Information

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

Problem Description

Suppose we have a class:

public class Foo { public void first() { print("first"); } public void second() { print("second"); } public void third() { print("third"); } }

The same instance of

Foo

will be passed to three different threads. Thread A will call

first()

, thread B will call

second()

, and thread C will call

third()

. Design a mechanism and modify the program to ensure that

second()

is executed after

first()

, and

third()

is executed after

second()

.

Note:

We do not know how the threads will be scheduled in the operating system, even though the numbers in the input seem to imply the ordering. The input format you see is mainly to ensure our tests' comprehensiveness.

Example 1:

Input:

nums = [1,2,3]

Output:

"firstsecondthird"

Explanation:

There are three threads being fired asynchronously. The input [1,2,3] means thread A calls first(), thread B calls second(), and thread C calls third(). "firstsecondthird" is the correct output.

Example 2:

Input:

nums = [1,3,2]

Output:

"firstsecondthird"

Explanation:

The input [1,3,2] means thread A calls first(), thread B calls third(), and thread C calls second(). "firstsecondthird" is the correct output.

Constraints:

nums

is a permutation of

[1, 2, 3]

.

Code Snippets

C++:

```
class Foo {
public:
Foo() {

}

void first(function<void()> printFirst) {

// printFirst() outputs "first". Do not change or remove this line.
printFirst();
}

void second(function<void()> printSecond) {

// printSecond() outputs "second". Do not change or remove this line.
printSecond();
}

void third(function<void()> printThird) {

// printThird() outputs "third". Do not change or remove this line.
printThird();
}
};
```

Java:

```
class Foo {

public Foo() {

}

public void first(Runnable printFirst) throws InterruptedException {

// printFirst.run() outputs "first". Do not change or remove this line.
printFirst.run();
}

public void second(Runnable printSecond) throws InterruptedException {

// printSecond.run() outputs "second". Do not change or remove this line.
printSecond.run();
}

public void third(Runnable printThird) throws InterruptedException {

// printThird.run() outputs "third". Do not change or remove this line.
printThird.run();
}
}
```

Python3:

```
class Foo:
def __init__(self):
pass
```

```
def first(self, printFirst: 'Callable[[], None]') -> None:

# printFirst() outputs "first". Do not change or remove this line.
printFirst()


def second(self, printSecond: 'Callable[[], None]') -> None:

# printSecond() outputs "second". Do not change or remove this line.
printSecond()


def third(self, printThird: 'Callable[[], None]') -> None:

# printThird() outputs "third". Do not change or remove this line.
printThird()
```

Python:

```
class Foo(object):
def __init__(self):
pass



def first(self, printFirst):
"""
:type printFirst: method
:rtype: void
"""

# printFirst() outputs "first". Do not change or remove this line.
printFirst()



def second(self, printSecond):
"""
:type printSecond: method
:rtype: void
"""

# printSecond() outputs "second". Do not change or remove this line.
```

```
printSecond()


def third(self, printThird):
"""
:type printThird: method
:rtype: void
"""

# printThird() outputs "third". Do not change or remove this line.
printThird()
```

C#:

```
public class Foo {

public Foo() {

}

public void First(Action printFirst) {

// printFirst() outputs "first". Do not change or remove this line.
printFirst();
}

public void Second(Action printSecond) {

// printSecond() outputs "second". Do not change or remove this line.
printSecond();
}

public void Third(Action printThird) {

// printThird() outputs "third". Do not change or remove this line.
printThird();
}
}
```

C:

```
typedef struct {
// User defined data may be declared here.

} Foo;

// Function Declaration, do not remove
void printFirst();
void printSecond();
void printThird();

Foo* fooCreate() {
Foo* obj = (Foo*) malloc(sizeof(Foo));

// Initialize user defined data here.

return obj;
}

void first(Foo* obj) {

// printFirst() outputs "first". Do not change or remove this line.
printFirst();
}

void second(Foo* obj) {

// printSecond() outputs "second". Do not change or remove this line.
printSecond();
}

void third(Foo* obj) {

// printThird() outputs "third". Do not change or remove this line.
printThird();
}

void fooFree(Foo* obj) {
// User defined data may be cleaned up here.

}
```

Solutions

C++ Solution:

```
/*
* Problem: Print in Order
* Difficulty: Easy
* Tags: general
*
* 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 Foo {
public:
Foo() {

}

void first(function<void()> printFirst) {

// printFirst() outputs "first". Do not change or remove this line.
printFirst();
}

void second(function<void()> printSecond) {

// printSecond() outputs "second". Do not change or remove this line.
printSecond();
}

void third(function<void()> printThird) {

// printThird() outputs "third". Do not change or remove this line.
printThird();
}
};
```

Java Solution:

```
/**
* Problem: Print in Order
* Difficulty: Easy
* Tags: general
*
* 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 Foo {

public Foo() {

}

public void first(Runnable printFirst) throws InterruptedException {

// printFirst.run() outputs "first". Do not change or remove this line.
printFirst.run();
}

public void second(Runnable printSecond) throws InterruptedException {

// printSecond.run() outputs "second". Do not change or remove this line.
printSecond.run();
}

public void third(Runnable printThird) throws InterruptedException {

// printThird.run() outputs "third". Do not change or remove this line.
printThird.run();
}
}
```

Python3 Solution:

```
"""
Problem: Print in Order
Difficulty: Easy
Tags: general

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 Foo:
def __init__(self):
pass


def first(self, printFirst: 'Callable[[], None]') -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Foo(object):
def __init__(self):
pass


def first(self, printFirst):
"""
:type printFirst: method
:rtype: void
"""

# printFirst() outputs "first". Do not change or remove this line.
printFirst()


def second(self, printSecond):
"""
:type printSecond: method
:rtype: void
"""

# printSecond() outputs "second". Do not change or remove this line.
printSecond()


def third(self, printThird):
```

```
"""
:type printThird: method
:rtype: void
"""

# printThird() outputs "third". Do not change or remove this line.
printThird()
```

C# Solution:

```
/*
* Problem: Print in Order
* Difficulty: Easy
* Tags: general
*
* 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 Foo {

public Foo() {

}

public void First(Action printFirst) {

// printFirst() outputs "first". Do not change or remove this line.
printFirst();
}

public void Second(Action printSecond) {

// printSecond() outputs "second". Do not change or remove this line.
printSecond();
}

public void Third(Action printThird) {

// printThird() outputs "third". Do not change or remove this line.
printThird();
}
}
```

C Solution:

```
/*
* Problem: Print in Order
* Difficulty: Easy
* Tags: general
*
* 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
*/

typedef struct {
// User defined data may be declared here.

} Foo;

// Function Declaration, do not remove
void printFirst();
void printSecond();
void printThird();

Foo* fooCreate() {
Foo* obj = (Foo*) malloc(sizeof(Foo));

// Initialize user defined data here.

return obj;
}

void first(Foo* obj) {

// printFirst() outputs "first". Do not change or remove this line.
printFirst();
}

void second(Foo* obj) {
```

```


// printSecond() outputs "second". Do not change or remove this line.
printSecond();
}

void third(Foo* obj) {

// printThird() outputs "third". Do not change or remove this line.
printThird();
}

void fooFree(Foo* obj) {
// User defined data may be cleaned up here.

}
```