Problem 1115: Print FooBar Alternately

Problem Information

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

Problem Description

Suppose you are given the following code:

class FooBar { public void foo() { for (int i = 0; i < n; i++) { print("foo"); } }

public void bar() { for (int i = 0; i < n; i++) { print("bar"); } } }

The same instance of

FooBar

will be passed to two different threads:

thread

A

will call

foo()

, while

thread

B

will call

bar()

.

Modify the given program to output

"foobar"

n

times.

Example 1:

Input:

n = 1

Output:

"foobar"

Explanation:

There are two threads being fired asynchronously. One of them calls foo(), while the other calls bar(). "foobar" is being output 1 time.

Example 2:

Input:

n = 2

Output:

"foobarfoobar"

Explanation:

"foobar" is being output 2 times.

Constraints:

1 <= n <= 1000

Code Snippets

C++:

```
class FooBar {
private:
int n;

public:
FooBar(int n) {
this->n = n;
}

void foo(function<void()> printFoo) {

for (int i = 0; i < n; i++) {

// printFoo() outputs "foo". Do not change or remove this line.
printFoo();
}
}

void bar(function<void()> printBar) {

for (int i = 0; i < n; i++) {

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

Java:

```
class FooBar {
private int n;

public FooBar(int n) {
this.n = n;
}

public void foo(Runnable printFoo) throws InterruptedException {

for (int i = 0; i < n; i++) {

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

public void bar(Runnable printBar) throws InterruptedException {

for (int i = 0; i < n; i++) {

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

Python3:

```
class FooBar:
def __init__(self, n):
self.n = n


def foo(self, printFoo: 'Callable[[], None]') -> None:

for i in range(self.n):

# printFoo() outputs "foo". Do not change or remove this line.
printFoo()
```

```
def bar(self, printBar: 'Callable[[], None]') -> None:

for i in range(self.n):

# printBar() outputs "bar". Do not change or remove this line.
printBar()
```

Python:

```
class FooBar(object):
def __init__(self, n):
self.n = n


def foo(self, printFoo):
"""
:type printFoo: method
:rtype: void
"""
for i in xrange(self.n):

# printFoo() outputs "foo". Do not change or remove this line.
printFoo()


def bar(self, printBar):
"""
:type printBar: method
:rtype: void
"""
for i in xrange(self.n):

# printBar() outputs "bar". Do not change or remove this line.
printBar()
```

C#:

```
public class FooBar {
private int n;

public FooBar(int n) {
this.n = n;
```

```
}

public void Foo(Action printFoo) {

for (int i = 0; i < n; i++) {

// printFoo() outputs "foo". Do not change or remove this line.
printFoo();
}
}

public void Bar(Action printBar) {

for (int i = 0; i < n; i++) {

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

C:

```
typedef struct {
int n;
} FooBar;

// Function declarations. Do not change or remove this line
void printFoo();
void printBar();

FooBar* fooBarCreate(int n) {
FooBar* obj = (FooBar*) malloc(sizeof(FooBar));
obj->n = n;
return obj;
}

void foo(FooBar* obj) {

for (int i = 0; i < obj->n; i++) {

// printFoo() outputs "foo". Do not change or remove this line.
```

```
printFoo();
}
}

void bar(FooBar* obj) {

for (int i = 0; i < obj->n; i++) {

// printBar() outputs "bar". Do not change or remove this line.
printBar();
}
}

void fooBarFree(FooBar* obj) {

}
```

Go:

```
type FooBar struct {
n int
}

func NewFooBar(n int) *FooBar {
return &FooBar{n: n}
}

func (fb *FooBar) Foo(printFoo func()) {
for i := 0; i < fb.n; i++ {
// printFoo() outputs "foo". Do not change or remove this line.
printFoo()
}
}

func (fb *FooBar) Bar(printBar func()) {
for i := 0; i < fb.n; i++ {
// printBar() outputs "bar". Do not change or remove this line.
printBar()
}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Print FooBar Alternately
* Difficulty: Medium
* 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 FooBar {
private:
int n;

public:
FooBar(int n) {
this->n = n;
}

void foo(function<void()> printFoo) {

for (int i = 0; i < n; i++) {

// printFoo() outputs "foo". Do not change or remove this line.
printFoo();
}
}

void bar(function<void()> printBar) {

for (int i = 0; i < n; i++) {

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

Java Solution:

```
/**
* Problem: Print FooBar Alternately
* Difficulty: Medium
* 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 FooBar {
private int n;

public FooBar(int n) {
this.n = n;
}

public void foo(Runnable printFoo) throws InterruptedException {

for (int i = 0; i < n; i++) {

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

public void bar(Runnable printBar) throws InterruptedException {

for (int i = 0; i < n; i++) {

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

Python3 Solution:

```
"""
Problem: Print FooBar Alternately
Difficulty: Medium
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 FooBar:
def __init__(self, n):
self.n = n


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

Python Solution:

```
class FooBar(object):
def __init__(self, n):
self.n = n


def foo(self, printFoo):
"""
:type printFoo: method
:rtype: void
"""
for i in xrange(self.n):

# printFoo() outputs "foo". Do not change or remove this line.
printFoo()


def bar(self, printBar):
"""
:type printBar: method
:rtype: void
"""
for i in xrange(self.n):

# printBar() outputs "bar". Do not change or remove this line.
```

```
printBar()
```

C# Solution:

```
/*
* Problem: Print FooBar Alternately
* Difficulty: Medium
* 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 FooBar {
private int n;

public FooBar(int n) {
this.n = n;
}

public void Foo(Action printFoo) {

for (int i = 0; i < n; i++) {

// printFoo() outputs "foo". Do not change or remove this line.
printFoo();
}
}

public void Bar(Action printBar) {

for (int i = 0; i < n; i++) {

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

C Solution:

```c
/*
* Problem: Print FooBar Alternately
* Difficulty: Medium
* 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 {
int n;
} FooBar;

// Function declarations. Do not change or remove this line
void printFoo();
void printBar();

FooBar* fooBarCreate(int n) {
FooBar* obj = (FooBar*) malloc(sizeof(FooBar));
obj->n = n;
return obj;
}

void foo(FooBar* obj) {

for (int i = 0; i < obj->n; i++) {

// printFoo() outputs "foo". Do not change or remove this line.
printFoo();
}
}

void bar(FooBar* obj) {

for (int i = 0; i < obj->n; i++) {

// printBar() outputs "bar". Do not change or remove this line.
printBar();
}
}

void fooBarFree(FooBar* obj) {
```

```
}
```

Go Solution:

```
// Problem: Print FooBar Alternately
// Difficulty: Medium
// 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

type FooBar struct {
n int
}

func NewFooBar(n int) *FooBar {
return &FooBar{n: n}
}

func (fb *FooBar) Foo(printFoo func()) {
for i := 0; i < fb.n; i++ {
// printFoo() outputs "foo". Do not change or remove this line.
printFoo()
}
}

func (fb *FooBar) Bar(printBar func()) {
for i := 0; i < fb.n; i++ {
// printBar() outputs "bar". Do not change or remove this line.
printBar()
}
}
```