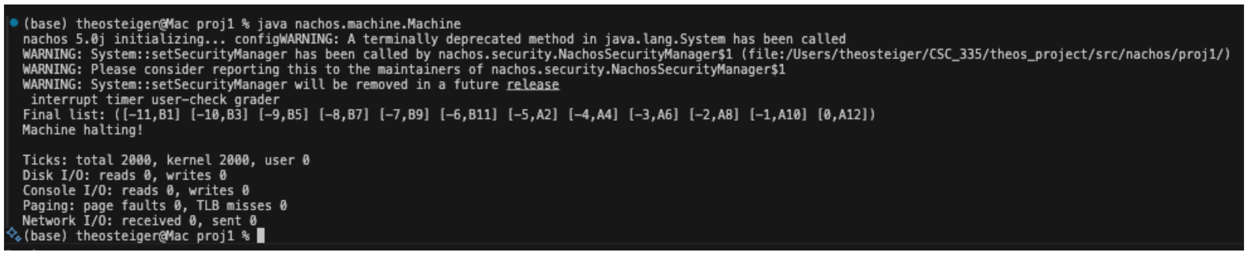
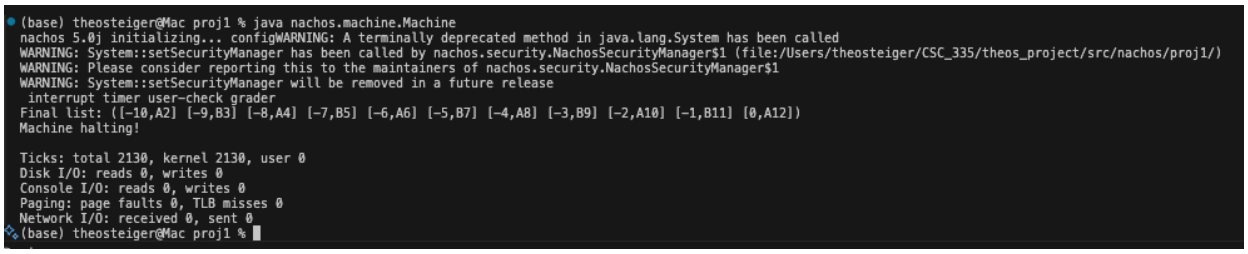
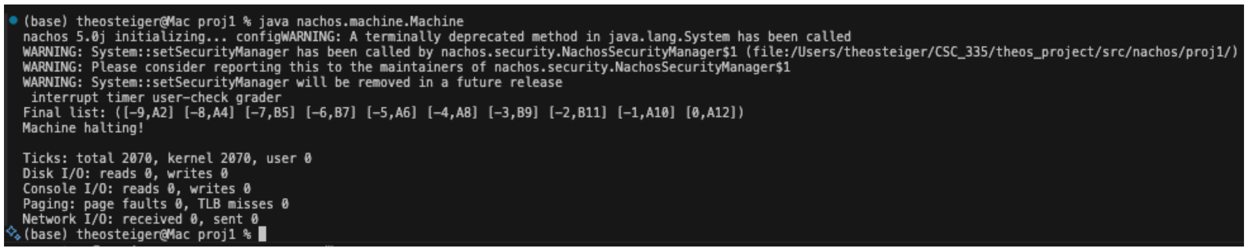
Three Screenshots from part 1:



```
(base) theosteiger@Mac proj1 % java nachos.machine.Machine
nachos 5.0j initializing... configWARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by nachos.security.NachosSecurityManager$1 (file:/Users/theosteiger/CSC_335/theos_project/src/nachos/proj1/)
WARNING: Please consider reporting this to the maintainers of nachos.security.NachosSecurityManager$1
WARNING: System::setSecurityManager will be removed in a future release
 interrupt timer user-check grader
Final list: ([-11,B1] [-10,B3] [-9,B5] [-8,B7] [-7,B9] [-6,B11] [-5,A2] [-4,A4] [-3,A6] [-2,A8] [-1,A10] [0,A12])
Machine halting!

Ticks: total 2000, kernel 2000, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: page faults 0, TLB misses 0
Network I/O: received 0, sent 0
(base) theosteiger@Mac proj1 %
```



```
(base) theosteiger@Mac proj1 % java nachos.machine.Machine
nachos 5.0j initializing... configWARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by nachos.security.NachosSecurityManager$1 (file:/Users/theosteiger/CSC_335/theos_project/src/nachos/proj1/)
WARNING: Please consider reporting this to the maintainers of nachos.security.NachosSecurityManager$1
WARNING: System::setSecurityManager will be removed in a future release
 interrupt timer user-check grader
Final list: ([-10,A2] [-9,B3] [-8,A4] [-7,B5] [-6,A6] [-5,B7] [-4,A8] [-3,B9] [-2,A10] [-1,B11] [0,A12])
Machine halting!

Ticks: total 2130, kernel 2130, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: page faults 0, TLB misses 0
Network I/O: received 0, sent 0
(base) theosteiger@Mac proj1 %
```



```
(base) theosteiger@Mac proj1 % java nachos.machine.Machine
nachos 5.0j initializing... configWARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by nachos.security.NachosSecurityManager$1 (file:/Users/theosteiger/CSC_335/theos_project/src/nachos/proj1/)
WARNING: Please consider reporting this to the maintainers of nachos.security.NachosSecurityManager$1
WARNING: System::setSecurityManager will be removed in a future release
 interrupt timer user-check grader
Final list: ([-9,A2] [-8,A4] [-7,B5] [-6,B7] [-5,A6] [-4,A8] [-3,B9] [-2,B11] [-1,A10] [0,A12])
Machine halting!

Ticks: total 2070, kernel 2070, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: page faults 0, TLB misses 0
Network I/O: received 0, sent 0
(base) theosteiger@Mac proj1 %
```

Part 2:
Fatal Error

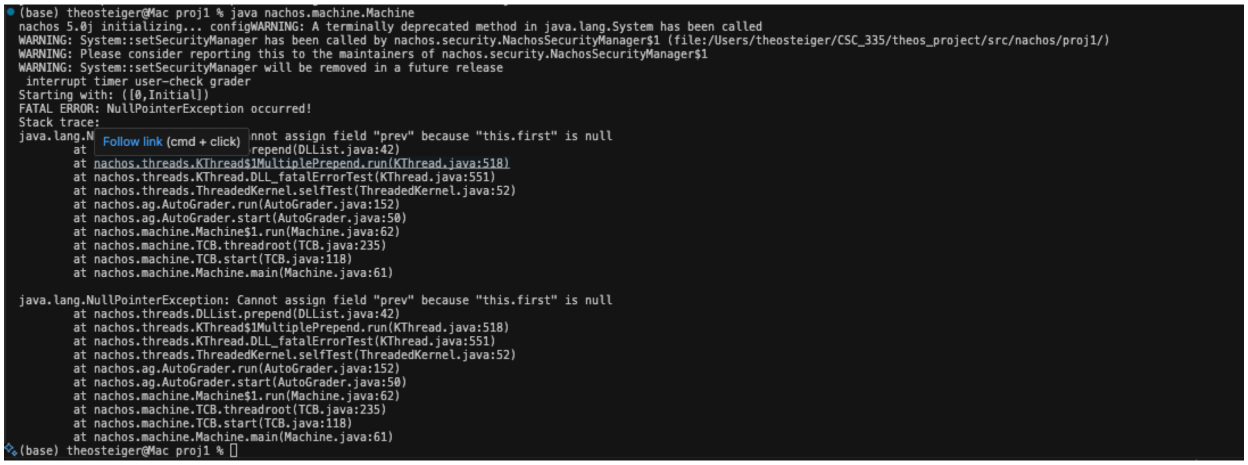
1. FATAL ERROR (NullPointerException)
 Initial State:
 - List contains one element: ([0,Initial])
 - Size: 1

 Threads and Method Calls:
 - Thread A (MultiplePrepend): Calls prepend("A1") then prepend("A2")
 - Thread B (RemoveAndAdd): Calls removeHead() then prepend("B-New")

 Critical Interleaving Sequence:
 - Thread A calls prepend("A1")
 - Thread A enters at location 0, sees first != null (pointing to "Initial")
 - Thread A calculates newKey = 0 - 1 = -1
 - Thread A creates newElement with data="A1" and key=-1 at line 32
 - Thread A enters non-empty branch
 - Thread A executes newElement.next = this.first (A1's next points to "Initial")
 - Thread A yields at location 4
 - Thread B calls removeHead()
 - Thread B reads first.data = "Initial"
 - Thread B executes this.first = first.next (first becomes null since "Initial" had no next)

- Thread B sets last = null (list is now empty)
- Thread B decrements size to 0
- Thread A resumes and tries to execute line 42: this.first.prev = newElement

CRASH: this.first is now null, causing NullPointerException when accessing .prev



```
(base) theosteiger@Mac proj1 % java nachos.machine.Machine
nachos 5.0j initializing... configWARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by nachos.security.NachosSecurityManager$1 (file:/Users/theosteiger/CSC_335/theos_project/src/nachos/proj1/)
WARNING: Please consider reporting this to the maintainers of nachos.security.NachosSecurityManager$1
WARNING: System::setSecurityManager will be removed in a future release
 interrupt timer user-check grader
Starting with: ([0,Initial])
FATAL ERROR: NullPointerException occurred!
Stack trace:
java.lang.N Follow link (cmd + click) nnot assign field "prev" because "this.first" is null
        at                           repend(DLList.java:42)
        at nachos.threads.KThread$1MultiplePrepend.run(KThread.java:518)
        at nachos.threads.KThread.DLL_fatalErrorTest(KThread.java:551)
        at nachos.threads.ThreadedKernel.selfTest(ThreadedKernel.java:52)
        at nachos.ag.AutoGrader.run(AutoGrader.java:152)
        at nachos.ag.AutoGrader.start(AutoGrader.java:50)
        at nachos.machine.Machine$1.run(Machine.java:62)
        at nachos.machine.TCB.threadroot(TCB.java:235)
        at nachos.machine.TCB.start(TCB.java:118)
        at nachos.machine.Machine.main(Machine.java:61)

java.lang.NullPointerException: Cannot assign field "prev" because "this.first" is null
        at nachos.threads.DLList.prepend(DLList.java:42)
        at nachos.threads.KThread$1MultiplePrepend.run(KThread.java:518)
        at nachos.threads.KThread.DLL_fatalErrorTest(KThread.java:551)
        at nachos.threads.ThreadedKernel.selfTest(ThreadedKernel.java:52)
        at nachos.ag.AutoGrader.run(AutoGrader.java:152)
        at nachos.ag.AutoGrader.start(AutoGrader.java:50)
        at nachos.machine.Machine$1.run(Machine.java:62)
        at nachos.machine.TCB.threadroot(TCB.java:235)
        at nachos.machine.TCB.start(TCB.java:118)
        at nachos.machine.Machine.main(Machine.java:61)
(base) theosteiger@Mac proj1 % 
```

corruption error:

2. NON-FATAL ERROR (List Corruption) - DLL_corruptionTest()

Initial State:
- Empty list: ()
- Size: 0

Threads and Method Calls:
- Thread A (MixedOperations): Calls prepend("A1"), then prepend("A2")
- Thread B (MixedOperations): Calls prepend("B1"), removeHead(), then prepend("B2")

Critical Interleaving Sequence:
- Thread A calls prepend("A1")
- Thread A enters at location 0, sees first == null
- Thread A calculates newKey = 0 at line 25
- Thread A yields at location 0
- Thread B calls prepend("B1")
- Thread B sees first == null (still empty)
- Thread B calculates newKey = 0 at line 25
- Thread B yields at location 1
- Thread A continues, creates newElement with data="A1" and key=0
- Thread A yields at location 2
- Thread B continues, creates newElement with data="B1" and key=0 (duplicate key!)
- Thread B completes prepend, setting first="B1", last="B1", size=1
- Thread A continues, enters empty branch (incorrectly, since B made it non-empty)
- Thread A sets first="A1", last="A1", size=2

- This overwrites B's node, leaving it orphaned

Actual Corrupted Result:
- List shows: ([-1,A2] [0,B2] [0,A1])
- CORRUPTION: Two elements with key 0 (duplicate keys)

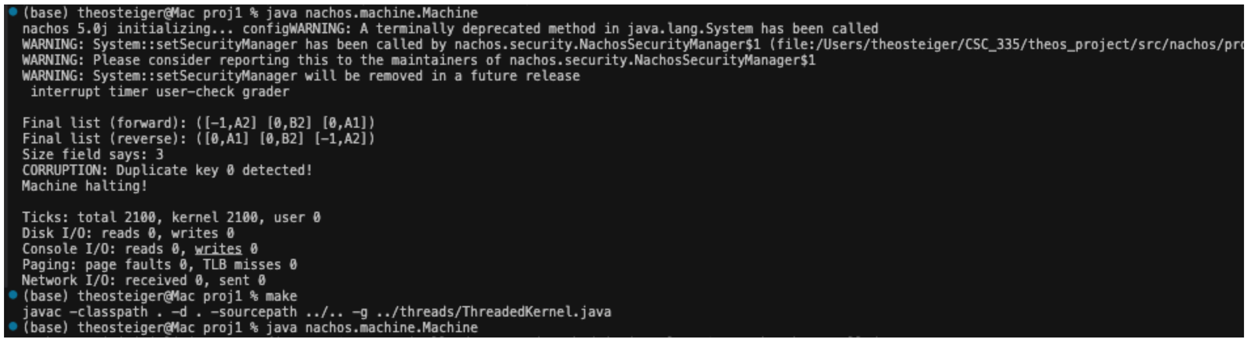
Expected Sequential Results:

If Thread A ran completely first, then Thread B:
1. A: prepend("A1") → ([0,A1]), size=1
2. A: prepend("A2") → ([-1,A2] [0,A1]), size=2
3. B: prepend("B1") → ([-2,B1] [-1,A2] [0,A1]), size=3
4. B: removeHead() → ([-1,A2] [0,A1]), size=2
5. B: prepend("B2") → ([-2,B2] [-1,A2] [0,A1]), size=3

Result: ([-2,B2] [-1,A2] [0,A1]) with unique keys

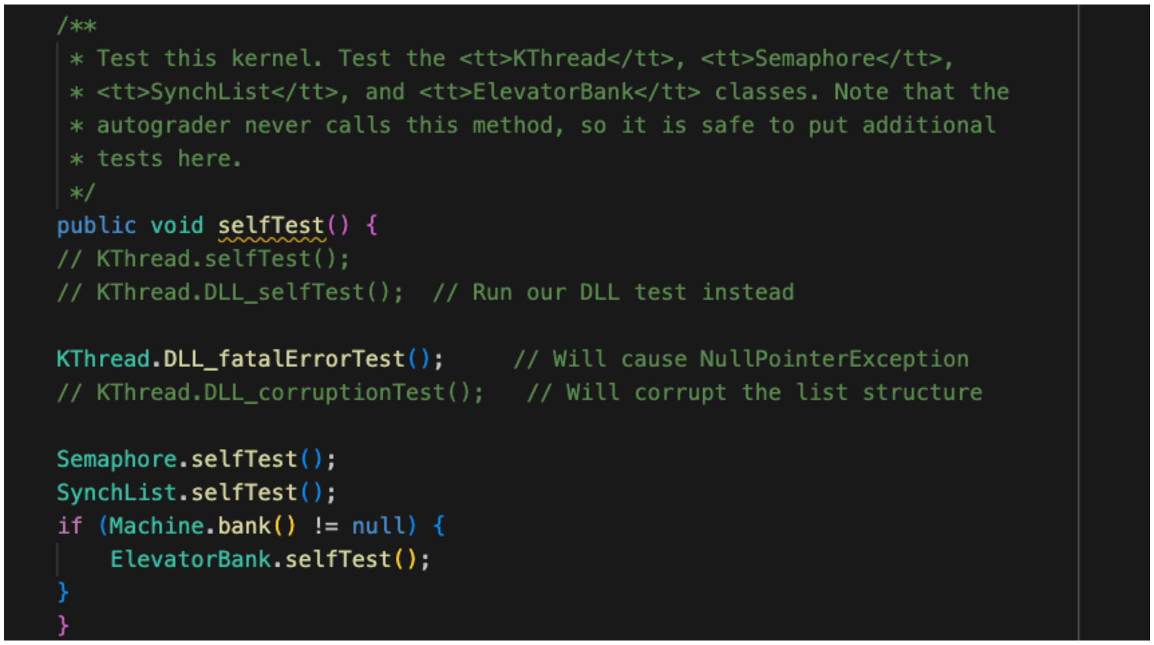
If Thread B ran completely first, then Thread A:
1. B: prepend("B1") → ([0,B1]), size=1
2. B: removeHead() → (), size=0
3. B: prepend("B2") → ([0,B2]), size=1
4. A: prepend("A1") → ([-1,A1] [0,B2]), size=2
5. A: prepend("A2") → ([-2,A2] [-1,A1] [0,B2]), size=3

Result: ([-2,A2] [-1,A1] [0,B2]) with **unique keys**

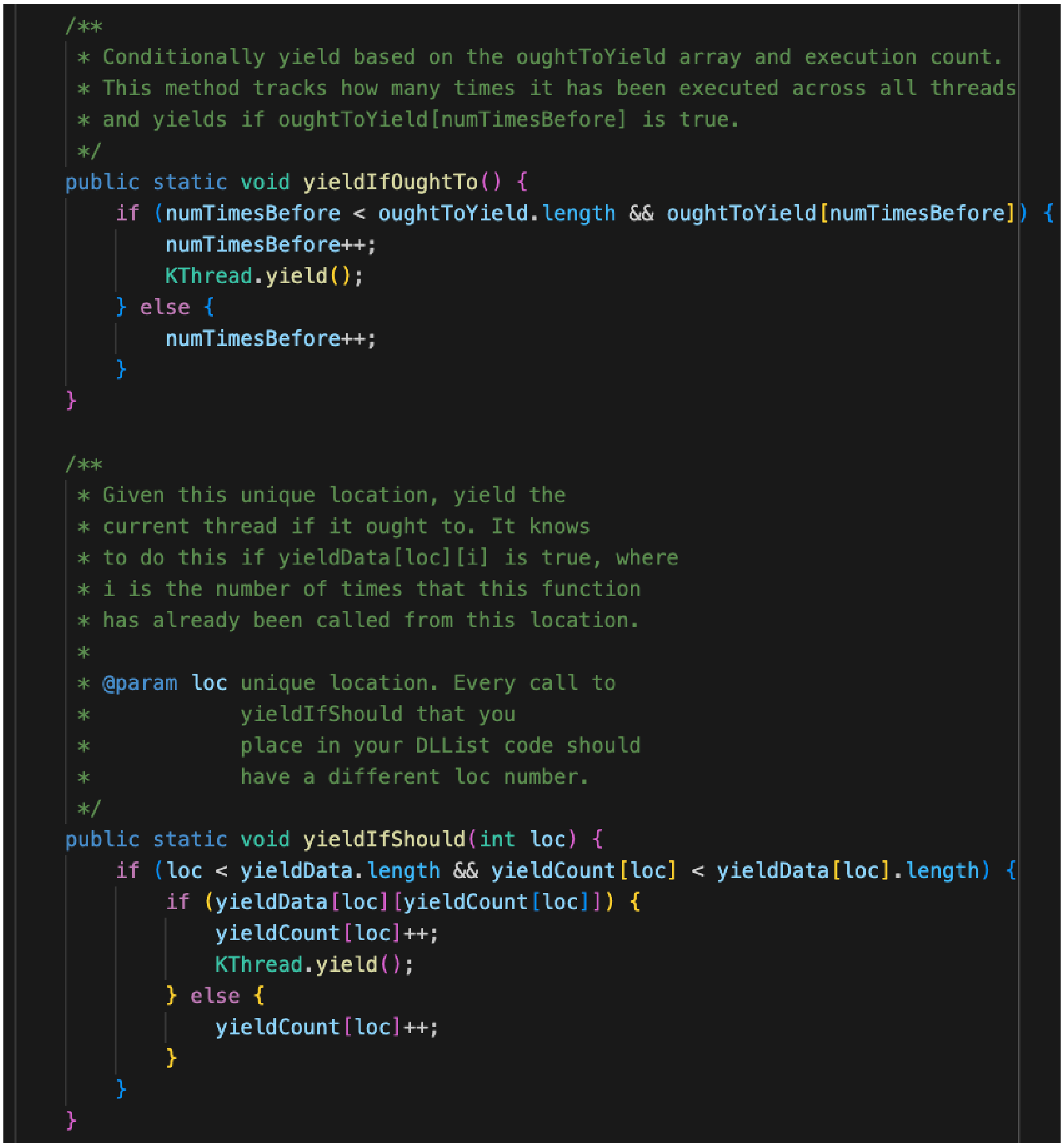


```
(base) theosteiger@Mac proj1 % java nachos.machine.Machine
nachos 5.0j initializing... configWARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by nachos.security.NachosSecurityManager$1 (file:/Users/theosteiger/CSC_335/theos_project/src/nachos/pr
WARNING: Please consider reporting this to the maintainers of nachos.security.NachosSecurityManager$1
WARNING: System::setSecurityManager will be removed in a future release
 interrupt timer user-check grader

Final list (forward): ([-1,A2] [0,B2] [0,A1])
Final list (reverse): ([0,A1] [0,B2] [-1,A2])
Size field says: 3
CORRUPTION: Duplicate key 0 detected!
Machine halting!

Ticks: total 2100, kernel 2100, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: page faults 0, TLB misses 0
Network I/O: received 0, sent 0
(base) theosteiger@Mac proj1 % make
javac -classpath . -d . -sourcepath ../.. -g ../threads/ThreadedKernel.java
(base) theosteiger@Mac proj1 % java nachos.machine.Machine
```

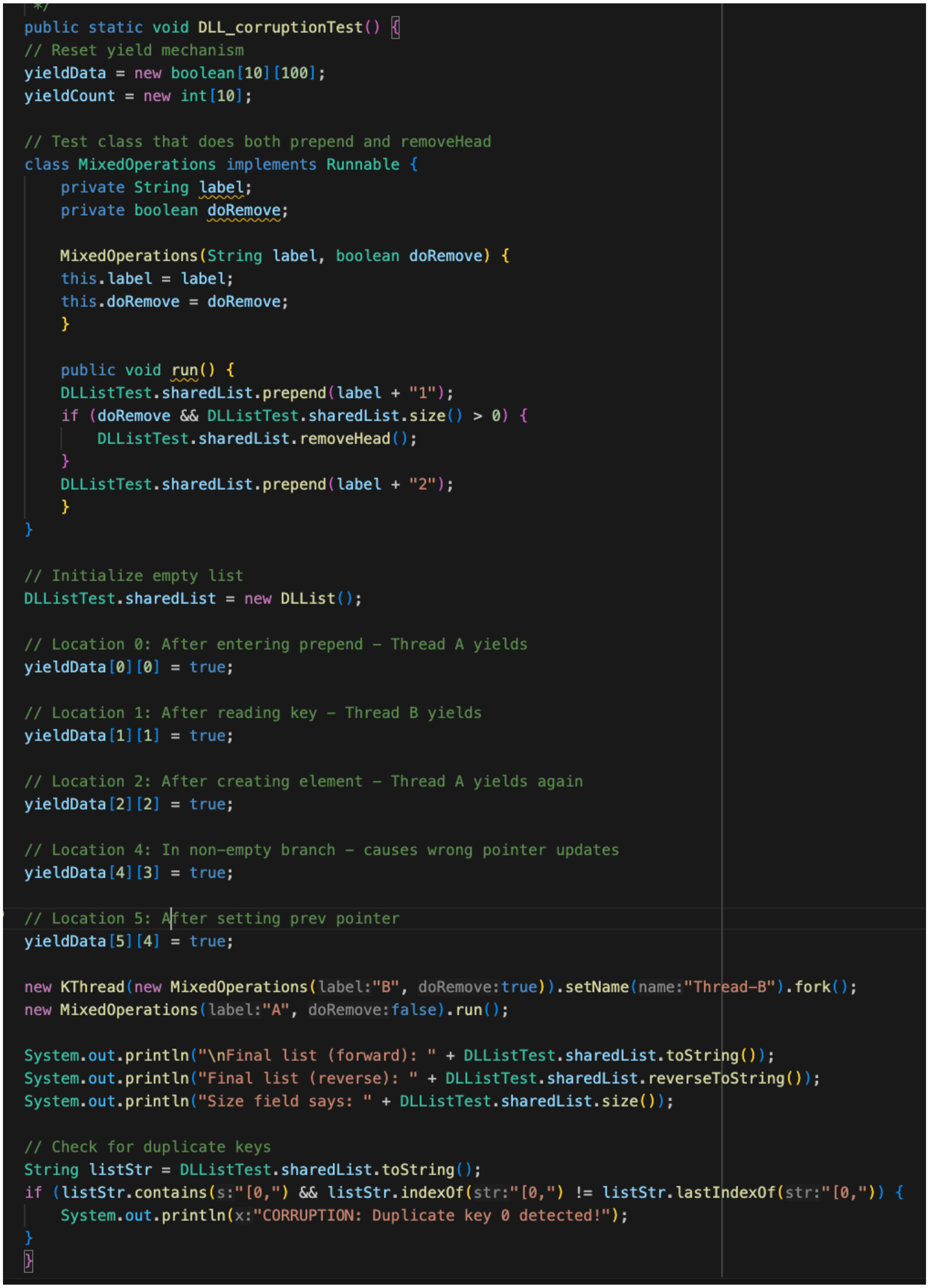
My methods:



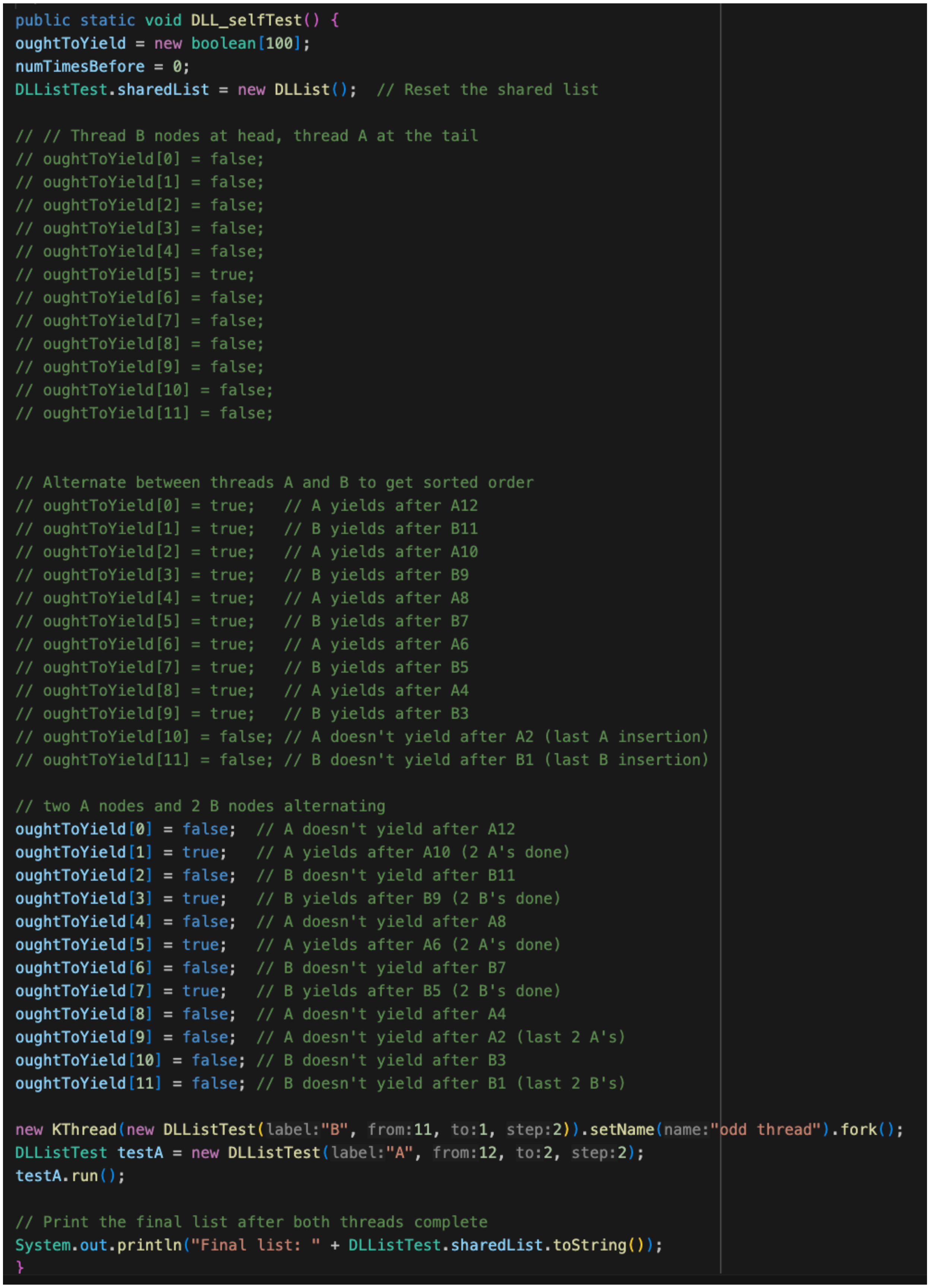
```
/**
 * Test this kernel. Test the <tt>KThread</tt>, <tt>Semaphore</tt>,
 * <tt>SynchList</tt>, and <tt>ElevatorBank</tt> classes. Note that the
 * autograder never calls this method, so it is safe to put additional
 * tests here.
 */
public void selfTest() {
// KThread.selfTest();
// KThread.DLL_selfTest();  // Run our DLL test instead

KThread.DLL_fatalErrorTest();     // Will cause NullPointerException
// KThread.DLL_corruptionTest();   // Will corrupt the list structure

Semaphore.selfTest();
SynchList.selfTest();
if (Machine.bank() != null) {
    ElevatorBank.selfTest();
}
}
```



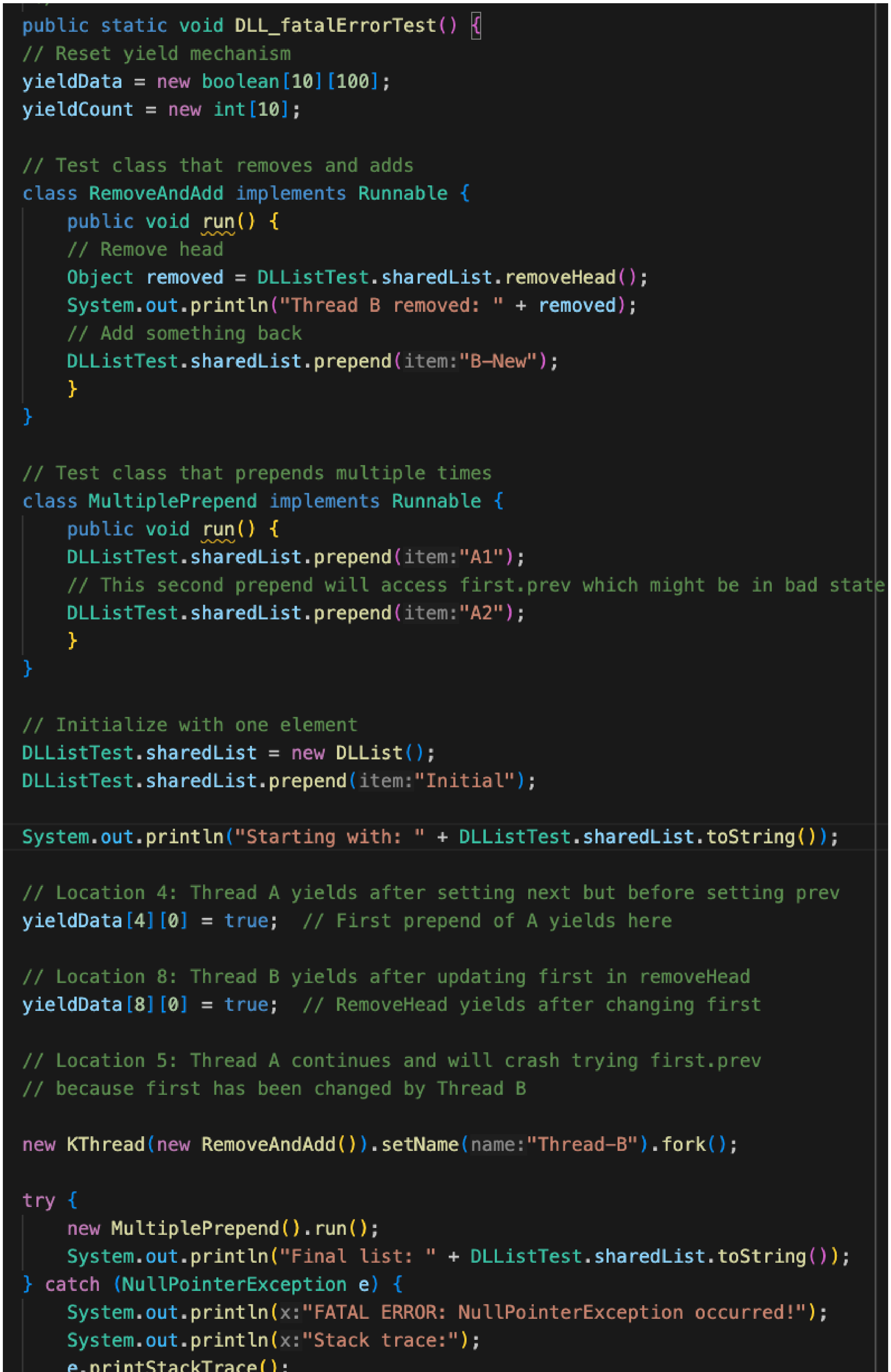
```java
/**
 * Conditionally yield based on the oughtToYield array and execution count.
 * This method tracks how many times it has been executed across all threads
 * and yields if oughtToYield[numTimesBefore] is true.
 */
public static void yieldIfOughtTo() {
    if (numTimesBefore < oughtToYield.length && oughtToYield[numTimesBefore]) {
        numTimesBefore++;
        KThread.yield();
    } else {
        numTimesBefore++;
    }
}

/**
 * Given this unique location, yield the
 * current thread if it ought to. It knows
 * to do this if yieldData[loc][i] is true, where
 * i is the number of times that this function
 * has already been called from this location.
 *
 * @param loc unique location. Every call to
 *            yieldIfShould that you
 *            place in your DLList code should
 *            have a different loc number.
 */
public static void yieldIfShould(int loc) {
    if (loc < yieldData.length && yieldCount[loc] < yieldData[loc].length) {
        if (yieldData[loc][yieldCount[loc]]) {
            yieldCount[loc]++;
            KThread.yield();
        } else {
            yieldCount[loc]++;
        }
    }
}
```



```
public static void DLL_corruptionTest() {
// Reset yield mechanism
yieldData = new boolean[10][100];
yieldCount = new int[10];

// Test class that does both prepend and removeHead
class MixedOperations implements Runnable {
    private String label;
    private boolean doRemove;

    MixedOperations(String label, boolean doRemove) {
    this.label = label;
    this.doRemove = doRemove;
    }

    public void run() {
    DLListTest.sharedList.prepend(label + "1");
    if (doRemove && DLListTest.sharedList.size() > 0) {
        DLListTest.sharedList.removeHead();
    }
    DLListTest.sharedList.prepend(label + "2");
    }
}

// Initialize empty list
DLListTest.sharedList = new DLList();

// Location 0: After entering prepend - Thread A yields
yieldData[0][0] = true;

// Location 1: After reading key - Thread B yields
yieldData[1][1] = true;

// Location 2: After creating element - Thread A yields again
yieldData[2][2] = true;

// Location 4: In non-empty branch - causes wrong pointer updates
yieldData[4][3] = true;

// Location 5: After setting prev pointer
yieldData[5][4] = true;

new KThread(new MixedOperations(label:"B", doRemove:true)).setName(name:"Thread-B").fork();
new MixedOperations(label:"A", doRemove:false).run();

System.out.println("\nFinal list (forward): " + DLListTest.sharedList.toString());
System.out.println("Final list (reverse): " + DLListTest.sharedList.reverseToString());
System.out.println("Size field says: " + DLListTest.sharedList.size());

// Check for duplicate keys
String listStr = DLListTest.sharedList.toString();
if (listStr.contains(s:"[0,") && listStr.indexOf(str:"[0,") != listStr.lastIndexOf(str:"[0,")) {
    System.out.println(x:"CORRUPTION: Duplicate key 0 detected!");
}
}
```



```java
public static void DLL_selfTest() {
oughtToYield = new boolean[100];
numTimesBefore = 0;
DLListTest.sharedList = new DLList();  // Reset the shared list

// // Thread B nodes at head, thread A at the tail
// oughtToYield[0] = false;
// oughtToYield[1] = false;
// oughtToYield[2] = false;
// oughtToYield[3] = false;
// oughtToYield[4] = false;
// oughtToYield[5] = true;
// oughtToYield[6] = false;
// oughtToYield[7] = false;
// oughtToYield[8] = false;
// oughtToYield[9] = false;
// oughtToYield[10] = false;
// oughtToYield[11] = false;


// Alternate between threads A and B to get sorted order
// oughtToYield[0] = true;   // A yields after A12
// oughtToYield[1] = true;   // B yields after B11
// oughtToYield[2] = true;   // A yields after A10
// oughtToYield[3] = true;   // B yields after B9
// oughtToYield[4] = true;   // A yields after A8
// oughtToYield[5] = true;   // B yields after B7
// oughtToYield[6] = true;   // A yields after A6
// oughtToYield[7] = true;   // B yields after B5
// oughtToYield[8] = true;   // A yields after A4
// oughtToYield[9] = true;   // B yields after B3
// oughtToYield[10] = false; // A doesn't yield after A2 (last A insertion)
// oughtToYield[11] = false; // B doesn't yield after B1 (last B insertion)

// two A nodes and 2 B nodes alternating
oughtToYield[0] = false;  // A doesn't yield after A12
oughtToYield[1] = true;   // A yields after A10 (2 A's done)
oughtToYield[2] = false;  // B doesn't yield after B11
oughtToYield[3] = true;   // B yields after B9 (2 B's done)
oughtToYield[4] = false;  // A doesn't yield after A8
oughtToYield[5] = true;   // A yields after A6 (2 A's done)
oughtToYield[6] = false;  // B doesn't yield after B7
oughtToYield[7] = true;   // B yields after B5 (2 B's done)
oughtToYield[8] = false;  // A doesn't yield after A4
oughtToYield[9] = false;  // A doesn't yield after A2 (last 2 A's)
oughtToYield[10] = false; // B doesn't yield after B3
oughtToYield[11] = false; // B doesn't yield after B1 (last 2 B's)

new KThread(new DLListTest(label:"B", from:11, to:1, step:2)).setName(name:"odd thread").fork();
DLListTest testA = new DLListTest(label:"A", from:12, to:2, step:2);
testA.run();

// Print the final list after both threads complete
System.out.println("Final list: " + DLListTest.sharedList.toString());
}
```



```
public static void DLL_fatalErrorTest() {
// Reset yield mechanism
yieldData = new boolean[10][100];
yieldCount = new int[10];

// Test class that removes and adds
class RemoveAndAdd implements Runnable {
    public void run() {
    // Remove head
    Object removed = DLListTest.sharedList.removeHead();
    System.out.println("Thread B removed: " + removed);
    // Add something back
    DLListTest.sharedList.prepend(item:"B-New");
    }
}

// Test class that prepends multiple times
class MultiplePrepend implements Runnable {
    public void run() {
    DLListTest.sharedList.prepend(item:"A1");
    // This second prepend will access first.prev which might be in bad state
    DLListTest.sharedList.prepend(item:"A2");
    }
}

// Initialize with one element
DLListTest.sharedList = new DLList();
DLListTest.sharedList.prepend(item:"Initial");

System.out.println("Starting with: " + DLListTest.sharedList.toString());

// Location 4: Thread A yields after setting next but before setting prev
yieldData[4][0] = true;  // First prepend of A yields here

// Location 8: Thread B yields after updating first in removeHead
yieldData[8][0] = true;  // RemoveHead yields after changing first

// Location 5: Thread A continues and will crash trying first.prev
// because first has been changed by Thread B

new KThread(new RemoveAndAdd()).setName(name:"Thread-B").fork();

try {
    new MultiplePrepend().run();
    System.out.println("Final list: " + DLListTest.sharedList.toString());
} catch (NullPointerException e) {
    System.out.println(x:"FATAL ERROR: NullPointerException occurred!");
    System.out.println(x:"Stack trace:");
    e.printStackTrace();
```