Working Interleavings:

```
Results in: B1, B3, B5 ..., A8, A10, A12
([-5,A2] [-4,A4] [-3,A6] [-2,A8] [-1,A10] [0,A12])
([-11,B1] [-10,B3] [-9,B5] [-8,B7] [-7,B9] [-6,B11] [-5,A2] [-4,A4] [-3,A6] [-2,A8] [-1,A10] [0,A12])


Results in: B1, A2, B3, ..., A10, B11, A12
([-11,B1] [-10,A2] [-9,B3] [-8,A4] [-7,B5] [-6,A6] [-5,B7] [-4,A8] [-3,B9] [-2,A10] [-1,B11] [0,A12])
([-11,B1] [-10,A2] [-9,B3] [-8,A4] [-7,B5] [-6,A6] [-5,B7] [-4,A8] [-3,B9] [-2,A10] [-1,B11] [0,A12])


Results in: B1, B3, A2, A4, ..., B11, A10, A12
([-11,B1] [-10,B3] [-9,A2] [-8,A4] [-7,B5] [-6,B7] [-5,A6] [-4,A8] [-3,B9] [-2,B11] [-1,A10] [0,A12])
([-11,B1] [-10,B3] [-9,A2] [-8,A4] [-7,B5] [-6,B7] [-5,A6] [-4,A8] [-3,B9] [-2,B11] [-1,A10] [0,A12])
```

First Working interleaving

(B1, B3, B5, B7, B9, B11, A2, A4, A6, A8, A10, A12)

`this.countDown(this.label, this.from, this.to, this.step);`
`System.err.println(myList);`
`KThread.yield();`
	`this.countDown(this.label, this.from, this.to, this.step);`
	`System.err.println(myList);`
	`KThread.yield();`

There is no interleaving between the threads in this scenario.each will run their loops to completion.

Second Working interleaving

(B1, A2, B3, A4, B5, A6, B7, A8, B9, A10, B11, A12)

```
public void prepend(Object item) {
       // If empty, start the list with the key = 0
       if (this.isEmpty()) {
           KThread.yieldIfShould(0); (false)
           DLLElement newNode = new DLLElement(item,
0);
           first = newNode;
           last = newNode;
       }
       // not empty, prepend with key = first.key - 1
       size +=1;
       KThread.yieldIfShould(2); (true)
```

```
public void prepend(Object item) {
       // If empty, start the list with the key = 0
       // not empty, prepend with key = first.key - 1
       else {
           KThread.yieldIfShould(1); (false)
           DLLElement newNode = new DLLElement(item,
first.key-1);
           newNode.next = first;
           first.prev = newNode;
           first = newNode;
       }
       size +=1;
       KThread.yieldIfShould(2); (true)
```

```
   } (NEXT ITERATION)
public void prepend(Object item) {
       // If empty, start the list with the key = 0
       // not empty, prepend with key = first.key - 1
       else {
           KThread.yieldIfShould(1); (false)
           DLLElement newNode = new DLLElement(item,
first.key-1);
           newNode.next = first;
           first.prev = newNode;
           first = newNode;
       }
       size +=1;
       KThread.yieldIfShould(2); (true)

}
```

```
}
```

Here, the interleaving is after each prepend completes.

Third Working interleaving

(B1, B3, A2, A4, B5, B7, A6, A8, B9, B11, A10, A12)

```
public void prepend(Object item) {
     // If empty, start the list with the key = 0
     if (this.isEmpty()) {
          KThread.yieldIfShould(0); (false)
          DLLElement newNode = new DLLElement(item, 0);
          first = newNode;
          last = newNode;
     }
     // not empty, prepend with key = first.key - 1
     size +=1;
     KThread.yieldIfShould(2); (false)
}(NEXT ITERATION)
public void prepend(Object item) {
     // If empty, start the list with the key = 0
     // not empty, prepend with key = first.key - 1
     else {
          KThread.yieldIfShould(1); (false)
          DLLElement newNode = new DLLElement(item,
first.key-1);
          newNode.next = first;
          first.prev = newNode;
          first = newNode;
     }
     size +=1;
     KThread.yieldIfShould(2); (true)

}
```

```
public void prepend(Object item) {
     // If empty, start the list with the key = 0
     // not empty, prepend with key = first.key - 1
     else {
          KThread.yieldIfShould(1); (false)
          DLLElement newNode = new DLLElement(item,
first.key-1);
          newNode.next = first;
          first.prev = newNode;
          first = newNode;
     }
     size +=1;
     KThread.yieldIfShould(2); (false)
}(NEXT ITERATION)
public void prepend(Object item) {
     // If empty, start the list with the key = 0
     // not empty, prepend with key = first.key - 1
     else {
          KThread.yieldIfShould(1); (false)
          DLLElement newNode = new DLLElement(item,
first.key-1);
          newNode.next = first;
          first.prev = newNode;
          first = newNode;
     }
     size +=1;
     KThread.yieldIfShould(2); (true)

}
```

Here it interleaves after every other prepend completes.

Broken Interleavings:

First Node Overwritten

```
public void prepend(Object item) {
     // If empty, start the list with the key = 0
     if (this.isEmpty()) {
         KThread.yieldIfShould(0); (true)



         DLLElement newNode = new DLLElement(item, 0);
         first = newNode;
         last = newNode;
     }
     // not empty, prepend with key = first.key - 1
     size +=1;
     KThread.yieldIfShould(2); (true)




}
```

```
public void prepend(Object item) {
     // If empty, start the list with the key = 0
     if (this.isEmpty()) {
         KThread.yieldIfShould(0); (true)



         DLLElement newNode = new DLLElement(item, 0);
         first = newNode;
         last = newNode;
     }
     // not empty, prepend with key = first.key - 1
     size +=1;
     KThread.yieldIfShould(2); (true)

}
```

```
Broken Interleaving: Overwrites first node
([-10,B1] [-9,A2] [-8,B3] [-7,A4] [-6,B5] [-5,A6] [-4,B7] [-3,A8] [-2,B9] [-1,A10] [0,B11])
([-10,B1] [-9,A2] [-8,B3] [-7,A4] [-6,B5] [-5,A6] [-4,B7] [-3,A8] [-2,B9] [-1,A10] [0,B11])
```

Here we can see this interleaving overwrites the first item added to the list, what should be A12. I have highlighted in gray the area that gets overwritten, and in yellow where said data gets overwritten.
Had this interleaving output a correct result, we would have seen an output identical to the working interleaving #2 (B1, A2, B3, A4, B5, A6, B7, A8, B9, A10, B11, A12)

Fatal Error

```
public void insert(Object item, Integer
sortKey) {
     DLLElement newNode = new
DLLElement(item, sortKey);
     // If list is empty, set first and last
to newnode and finish
     if (this.isEmpty()) {
          size += 1;
          last = newNode;
          KThread.yieldIfShould(3);









          first = newNode;
          return;
     }
}
```

```
public Object removeHead() {
     if (this.isEmpty()) return null;(false)
     Object toReturn = first.data;
     first = first.next;
     if (first == null) last = null;
     else first.prev = null;
     size -= 1;
     return toReturn;
}
```

```
Broken Interleaving: Results in Null Pointer

java.lang.NullPointerException: Cannot read field "data" because "this.first" is null
        at nachos.threads.DLList.removeHead(DLList.java:59)
        at nachos.threads.KThread$DLListBadTest.run(KThread.java:496)
        at nachos.threads.KThread.DLL_BadTest(KThread.java:591)
        at nachos.threads.KThread.selfTest(KThread.java:544)
        at nachos.threads.ThreadedKernel.selfTest(ThreadedKernel.java:49)
        at nachos.ag.AutoGrader.run(AutoGrader.java:152)
        at nachos.ag.AutoGrader.start(AutoGrader.java:50)
        at nachos.machine.Machine$1.run(Machine.java:63)
        at nachos.machine.TCB.threadroot(TCB.java:235)
        at nachos.machine.TCB.start(TCB.java:118)
        at nachos.machine.Machine.main(Machine.java:62)
```

This interleaving attempts to insert a node and remove a node at the same time. Each run without interleaving would not result in any errors, but under these interleaving circumstances it results in a Null Pointer Exception, as the removeHead attempts to remove a node from first, when first has not yet had a node put into it.L