

Task01 Explanation: this task is very easy I just have to Start from the tail and move towards the head and no further explanation needed

Task01 code:

```
public void printReverse() {  
    DLLNode<T> tmp = tail;  
    while (tmp != null) {  
        System.out.print(tmp.info + " ");  
        tmp = tmp.prev;  
    }  
    System.out.println();  
}
```

Task01 output:

```
f2ea\redhat.java (jdk_ws  
initail List:  
a0 a1 a2 a3 a4  
reversed list:  
a4 a3 a2 a1 a0  
PS C:\Users\seahd\OneDr
```

```
f2ea\redhat.java (jdk  
Initial List:  
1 2 3 4 5  
Reversed List:  
5 4 3 2 1  
PS C:\Users\seahd\On
```

```
f2ea\redhat.java  
Initial List:  
A B C D E  
Reversed List:  
E D C B A  
PS C:\Users\seahd\On
```

Task02 explanation: first I Initialize a count to keep track of the nodes visited and I Initialize a flag to indicate the traversal direction (forward or backward) then I Loop until the count reaches 7 (to find the seventh node) and after all of this I update the next pointer of the previous node and update the previous pointer of the next node

Some of Task02 code:

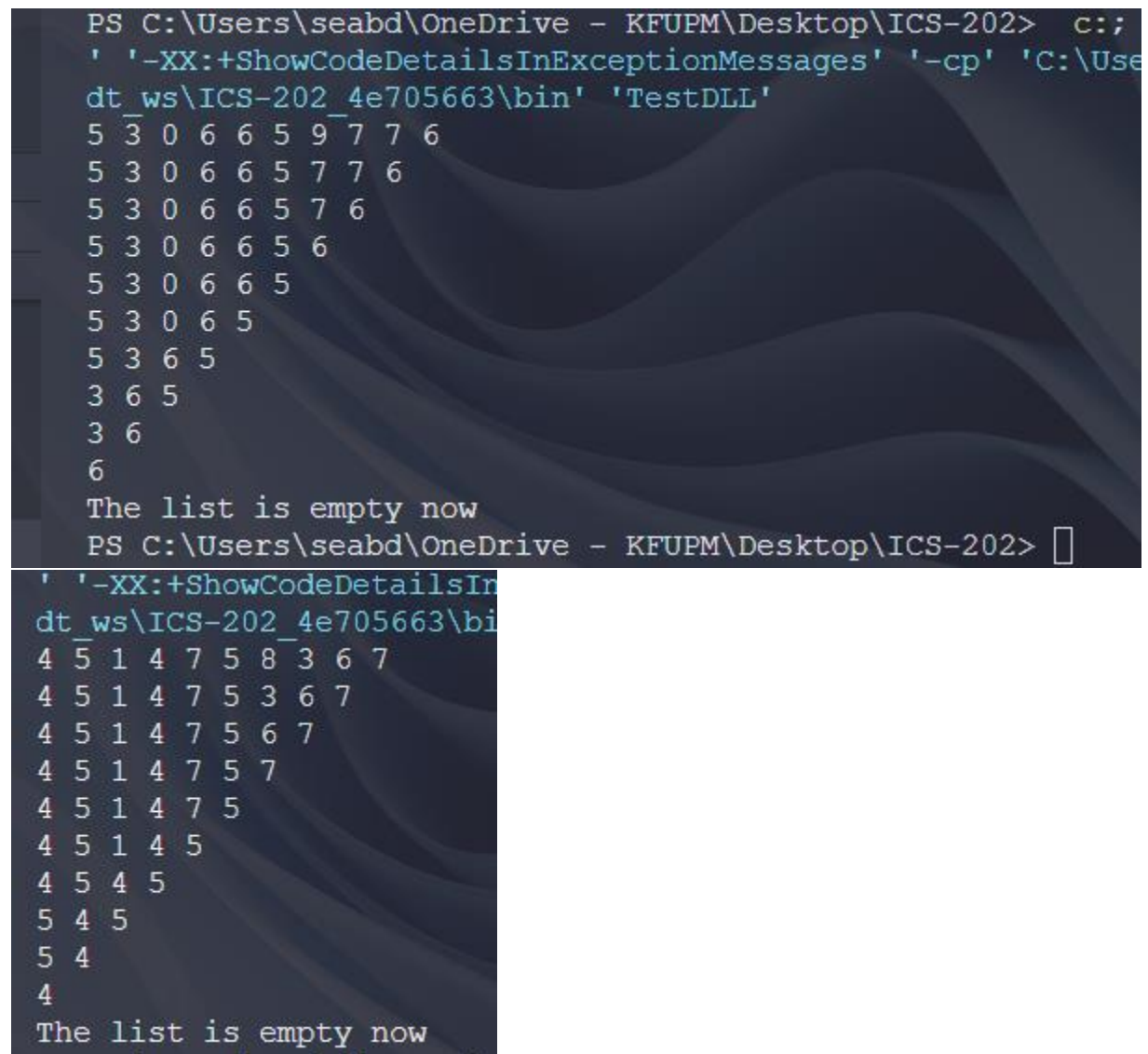
```
while (count < 7) {  
    if (forward) {  
        if (current.next != null) {  
            current = current.next;  
        } else {  
            forward = false;  
            if (current != null) {  
                current = current.prev;  
            }  
        }  
    } else {  
        if (current.prev != null) {  
            current = current.prev;  
        } else {  
            forward = true;  
            if (current != null) {  
                current = current.next;  
            }  
        }  
    }  
    count++;  
}  
  
if (current != null) {
```

```

if (current == head) {
    deleteFromHead();
} else if (current == tail) {
    deleteFromTail();
} else {
    current.prev.next = current.next;
    current.next.prev = current.prev; }}

```

Task02 output:



```

PS C:\Users\seabd\OneDrive - KFUPM\Desktop\ICS-202> c::;
' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Use
dt_ws\ICS-202_4e705663\bin' 'TestDLL'
5 3 0 6 6 5 9 7 7 6
5 3 0 6 6 5 7 7 6
5 3 0 6 6 5 7 6
5 3 0 6 6 5 6
5 3 0 6 6 5
5 3 0 6 5
5 3 6 5
3 6 5
3 6
6
The list is empty now
PS C:\Users\seabd\OneDrive - KFUPM\Desktop\ICS-202> 
' '-XX:+ShowCodeDetailsIn
dt_ws\ICS-202_4e705663\bi
4 5 1 4 7 5 8 3 6 7
4 5 1 4 7 5 3 6 7
4 5 1 4 7 5 6 7
4 5 1 4 7 5 7
4 5 1 4 7 5
4 5 1 4 5
4 5 4 5
5 4 5
5 4
4
The list is empty now

```

```

dt_ws\ICS-202_4e705663\b
1 8 2 3 8 2 7 4 7 2
1 8 2 3 8 2 4 7 2
1 8 2 3 8 2 7 2
1 8 2 3 8 2 2
1 8 2 3 8 2
1 8 2 3 2
1 8 3 2
8 3 2
8 3
3
The list is empty now
PS C:\Users\seahd\OnePri

```

Task03 explanation: first , I save references to next nodes in both lists then I Insert the element from newList after the current element in this list then I Move to the next elements in both lists and if one list has extra elements, append them to the end of the invoking list by attach the remaining elements from newList to the end of this list and update the tail to the new tail.

Some of Task03 code:

```

while (currentThis != null && currentNew != null) {
    // Save references to next nodes in both lists
    DLLNode<T> nextThis = currentThis.next;
    DLLNode<T> nextNew = currentNew.next;
    currentThis.next = currentNew;
    currentNew.prev = currentThis;
    currentNew.next = nextThis;
    currentThis = nextThis;
    currentNew = nextNew }
if (currentNew != null) {
    tail.next = currentNew;
    currentNew.prev = tail;
    tail = newList.tail; } }

```

Task03 output:

```
\java.exe' '-XX:+ShowCodeDetails
f2ea\redhat.java\jdt_ws\ICS-202_
List 1:
1 2 3 4
List 2:
5 6 7 8
After inserting alternately:
1 5 2 6 3 7 4 8
```

```
\java.exe' '-XX:+ShowCodeDetails
f2ea\redhat.java\jdt_ws\ICS-202_
List 1:
A B C
List 2:
X Y Z
After inserting alternately:
A X B Y C Z
```

```
f2ea\redhat.java\jdt_ws\ICS-20
List 1:
A B C D E
List 2:
X Y Z
After inserting alternately:
A X B Y C Z D E
PS C:\Users\seabd\OneDrive G-o K
```