KEY: Orange highlight = Test Added during this milestone Red highlight = Impossible test case

Test	Equivalence class part	Equivalence class partitioning(Wect, Sect, Wrect, Srect)					
testAdd	<ul> <li>Variables: <ul> <li>List length L</li> <li>Object O</li> </ul> </li> <li>Equivalence classes <ul> <li>L1:L=0</li> <li>L2:L&gt;0</li> <li>O1: O does not already exist in list</li> <li>O2: O already exists in list</li> </ul> </li> <li>Wect</li> </ul>						
	ID	L	0				
	WECT1	L1	01				
	WECT2	L2	O2				
	• <u>Sect</u>						
	ID	L	I				
	SECT1	L1	01				
	SECT2	L2	01				
	SECT3	L1	O2				
	SECT4	L2	O2				
			< 0 but this is not potesting of a null varia				
testClear	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and rect are covered by testing of a null variable list</li> </ul> </li> </ul>						
testContains	Variables:						

List length L   Object O		1							
ID		<ul> <li>○ Object O</li> <li>● Equivalence classes</li> <li>○ L1:L=0</li> <li>○ L2:L&gt;0</li> <li>○ O1:O is in list</li> <li>○ O2:O is not in list</li> </ul>							
WECT2				L	0				
Sect   ID			WECT1	L1	O1				
ID			WECT2	L2	O2				
SECT2		• 5	Sect			•			
SECT2			ID	L	I				
SECT3			SECT1	L1	01				
SECT4  • Wrect and Srect would involve L < 0 but this is not possible • Wrect and srect are covered by WECT1, SECT1 • Wrect and srect are also covered by testing of a null variable list  • Variables:  • List length L  • Object list O  • Equivalence classes  • L1: L=0  • L2: L>0  • O1: All of O is in list  • O2: Not all of O is not in list  • Wect  ID  WECT1  L1  O1  WECT2  L2  O2  • Sect  ID  L  I			SECT2	L2	O1				
Wrect and Srect would involve L < 0 but this is not possible Wrect and srect are covered by WECT1, SECT1 Wrect and srect are also covered by testing of a null variable list  Variables:  List length L Object list O Equivalence classes L1: L=0 L2: L>0 O1: All of O is in list O2: Not all of O is not in list Wect  D WECT1 L1 O1 WECT2 L2 O2  Sect  D L I  U  U  U  U  U  U  U  U  U  U  U  U			O2						
Wrect and srect are covered by WECT1, SECT1  Wrect and srect are also covered by testing of a null variable list  Variables:  List length L Object list O  Equivalence classes  L1: L=0 L2: L>0 O1: All of O is in list O2: Not all of O is not in list  Wect  ID  WECT1  L1  O1  WECT2  Sect  ID  L  I		SECT4 L2 O2							
<ul> <li>List length L</li> <li>Object list O</li> <li>Equivalence classes</li> <li>L1: L=0</li> <li>L2: L&gt;0</li> <li>O1: All of O is in list</li> <li>O2: Not all of O is not in list</li> <li>Wect</li> <li>ID</li> <li>WECT1</li> <li>L1</li> <li>O1</li> <li>WECT2</li> <li>L2</li> <li>O2</li> <li>Sect</li> <li>ID</li> <li>L</li> <li>I</li> &lt;</ul>		• \	Nrect and srect are Nrect and srect are	covered by WE	ECT1, SECT1				
WECT1 L1 O1 WECT2 L2 O2  Sect ID L I	testContainsAll	• [	<ul> <li>List length L</li> <li>Object list O</li> <li>Equivalence classes</li> <li>L1: L=0</li> <li>L2: L&gt;0</li> <li>O1: All of O is</li> <li>O2: Not all of</li> </ul>	s in list					
WECT2 L2 O2  • Sect  ID			ID	L	0				
• Sect ID L I			WECT1	L1	01				
ID L I			WECT2	L2	O2				
<u> </u>		• }	Sect		1				
SECT1 L1 O1			ID	L	1				
			SECT1	L1	O1				

		SECT2	L2	01		
		SECT3	L1	O2		
		SECT4	L2	O2		
		Wrect and Srect wou Wrect and srect are variable list				
testEquals		Variables:      List length L Equivalence classes     L1:L=0     L2:L>0 Thus, wect and sect and a nonempty     This is alread Wrect and srect are	are both cover ly done lld involve L < (	) but this is no	t possible	
testHashCode	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1:L=0</li> <li>L2:L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and srect are covered by testing of a null variable list</li> </ul> </li> </ul>					
testGet	•	Variables:  o List length L o Index I  Equivalence classes o L1: L=0 o L2: L>0 o I1: I<0 o I2: I>=0&&I <l i="" i3:="" o="">=L</l>				
		ID	L	I	]	
		WECT1	L1	l1		
		WECT2	L2	12	]	

					_		
		WECT3		L1	13		
	•	Sect					
		ID		L	1		
		SECT1		L1	I1		
		SECT2		L2	I1		
		SECT3		L1	12		
		SECT4		L2	12		
		SECT5		L1	13		
		SECT6		L2	13		
testIndexOf	<ul> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and srect are covered by WECT1, WECT3, SECT1, SECT2, SECT5, and SECT6</li> <li>Wrect and srect are also covered by testing of a null variable list</li> <li>Variables:         <ul> <li>List length L</li> <li>Object O</li> </ul> </li> <li>Equivalence classes         <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> <li>O1: O is in list</li> </ul> </li> </ul>						
	• Wect	o O2: O is not	in list				
	ID		L		0		
	WECT	1	L1		O1		
	WECT	<sup>-</sup> 2	L2		O2		
	• Sect						
	ID		L		I		
	SECT	1	L1		01		

				1			
	SECT2	L2	01				
	SECT3	L1	O2				
	SECT4	L2	O2				
	<ul><li>Wrect and Srect wor</li><li>Wrect and srect are</li><li>Wrect and srect are</li></ul>	covered by WECT	1, SECT1				
testIsEmpty	and a nonempt ○ This is a • Wrect and Sred	asses sect are both cove	0 but this is no	t possible			
testIterator	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>This is already done</li></ul></li></ul>						
testRemove	and a nonempt  ○ This is a  • Wrect and Sred	asses sect are both cove	0 but this is no	t possible			
testRemoveAll	<ul> <li>Variables:         <ul> <li>List leng</li> </ul> </li> <li>Equivalence classification</li> <li>L1: L=0</li> </ul>						

<ul> <li>L2: L&gt;0</li> <li>Thus, wect and sect are both covered by using an empty and a nonempty         <ul> <li>This is already done</li> </ul> </li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible.</li> <li>Wrect and rect are covered by testing of a null variable list testRemoveByIndex</li> <li>Variables:         <ul> <li>List length L</li> <li>Index I</li> </ul> </li> <li>Equivalence classes         <ul> <li>L1:L=0</li> <li>L2:L&gt;0</li> <li>I1:I&lt;0</li> <li>I2:I&gt;=0 &amp;&amp; I</li> </ul> </li> <li>I3:I&gt;=L</li> </ul>						
	• Wect	-L	<u> </u>			
	ID	L	1			
	WECT1	L1	I1			
	WECT2	L2	12			
	WECT3	L1	13			
	• Sect	1	<b>-</b>			
	ID	L	1			
	SECT1	L1	I1			
	SECT2	L2	I1			
	SECT3	L1	12			
	SECT4	L2	12			
	SECT5	L1	13			
	SECT6	L2	13			
	Wrect and s     SECT2, SEC	rect are covered CT5, and SECT6	ve L < 0 but this is by WECT1, WEC	CT3, SECT1,		
testRetainAll	Variables:	ı L				

	<ul> <li>List of objects O</li> <li>Equivalence classes</li> <li>L1:L=0</li> <li>L2:L&gt;0</li> <li>O1: All elements of O exist in list</li> <li>O2: Some elements of O exist in list</li> <li>O3: No elements of O exist in list</li> <li>Wect</li> </ul>								
		ID L O							
		WECT1	L1	(	O1				
		WECT2	L2	(	O2				
		WECT3	L1	(	O3				
	• Sect								
		ID	L		I				
		SECT1	L1		O1				
		SECT2	L2		O1				
		SECT3	L1		O2				
		SECT4	L2		O2				
		SECT5	L1		О3				
		SECT6	L2		O3				
	• V	/rect and Srect /rect and srect a ariable list							
testSet	variable list								
		D	L	1	0				

		WECT1	L1	I1	01			
		WECT2	L2	12	O2			
		WECT3	L1	13	01			
	• Sect	• Sect						
		ID	L	1	0			
		SECT1	L1	I1	O1			
		SECT2	L2	l1	O1			
		SECT3	L1	12	01			
		SECT4	L2	12	01			
		SECT5	L1	13	01			
		SECT6	L2	13	01			
		SECT7	L1	l1	O2			
		SECT8	L2	I1	O2			
		SECT9	L1	12	O2			
		SECT10	L2	12	O2			
		SECT11	L1	13	O2			
		SECT12	L2	13	O2			
		ct and Srect would ct and srect are c						
testSublist	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li></ul></li></ul>							
testSubListAddBegin		Variables: ○ List lengt Equivalence clas						

	<ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty</li> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and Srect are covered by testing of a null variable list</li> </ul>
testSubListAddEnd	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>This is already done</li></ul></li></ul>
testSubListAddMiddle	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>This is already done</li></ul></li></ul>
testSubListRemove	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <li>L2: L&gt;0</li> </ul> </li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>Wrect and Srect are covered by testing of a null variable list</li> <li>This is already done</li> <li>This is already done</li></ul></li></ul>
testToArray	<ul> <li>Variables:         <ul> <li>List length L</li> </ul> </li> <li>Equivalence classes             <ul> <li>L1: L=0</li> <ul> <li>L2: L&gt;0</li> </ul> </ul></li> <li>Thus, wect and sect are both covered by using an empty list and a nonempty                     <ul> <li>This is already done</li> <li>Wrect and Srect would involve L &lt; 0 but this is not possible</li> </ul> </li> </ul>

• Wrect and Srect are covered by testing of a null variable list

Test	Boundary Value Analy	rsis					
testSet	<ul> <li>Variables: <ul> <li>List length L</li> <li>Index I</li> <li>Object O</li> </ul> </li> <li>Test Set <ul> <li>BVL:{nom:L=0,nom:L&gt;0}</li> <li>BVI:{min-:I=-1,min:I=0,min+:I=1,nom:1<i<l,max-: -="" 1,="" 2,="" i="L}&lt;/li" max+:="" max:=""> <li>BVO: {nom: O can be inserted, nom: O cannot be inserted}</li> </i<l,max-:></li></ul> </li> <li>BVA (basic)</li> </ul>						
	ID	L	1	0			
	BVA (basic) 1	L>0	0	O can be inserted			
	BVA (basic) 2	L>0	1	O can be inserted			
	BVA (basic) 3	L>0	nom	O can be inserted			
	BVA (basic) 4	L>0	L-2	O can be inserted			
	BVA (basic) 5	L>0	L-1	O can be inserted			
	BVA (robust)	-					
	ID	L	ı	0			
	BVA (robust) 1	L>0	-1	O can be inserted			
	BVA (robust) 2	L>0	0	O can be inserted			
	BVA (robust) 3	L>0	1	O can be inserted			
	BVA (robust) 4	L>0	nom	O can be inserted			

	E	BVA (robust) 5	L>0		L-2	O can be inserted	
	E	BVA (robust) 6	L>0		L-1	O can be inserted	
	E	3VA (robust) 7	L>0		L	O can be inserted	
testGet	•		m:L= n-:l=-	1,mi	n:I=0,mir	n+:I=1,nom:1< - 1, max+: I =	
		ID		L		I	
		BVA (basic) 1		L>(	)	0	
		BVA (basic) 2		L>(	)	1	
		BVA (basic) 3		L>(	)	nom	
		BVA (basic) 4		L>(	)	L-2	
		BVA (basic) 5		L>	0	L - 1	
	•	BVA (robust)					
		ID		L		I	
		BVA (robust) 1		L>0	)	-1	
		BVA (robust) 2		L>0	)	0	
		BVA (robust) 3		L>0	)	1	
		BVA (robust) 4		L>0	)	nom	
		BVA (robust) 5		L>0	)	L-2	
		BVA (robust) 6		L>0	)	L-1	
		BVA (robust) 7		L>0	)	L	

Test	Decision Table						
testAdd	<ul> <li>Conditions <ul> <li>C1:L&lt;=0?</li> <li>C2: Object can be inserted?</li> </ul> </li> <li>Actions <ul> <li>A1: Object is inserted</li> </ul> </li> <li>Table</li> </ul>						
	Test Case ID	C1	C2	Expected Output			
	TC1	Y/T	-	-			
	TC2	N/F	Y/T	Object inserted	1		
	TC3	N/F	N/F	No change			
					_		
testContains	○ C2 • Actions	s 1:L<=0? 2: Object 1: Object		list			
	Test Case ID	C1	C2	Expected Output			
	TC1	Y/T	-	-			
	TC2	N/F	Y/T	True			
	TC3	N/F	N/F	False			
testContainsAll	Conditions						
	TC2	N/F	Y/T	True			

	TC3	N/F	N/F	False		
				'		
testEquals	<ul> <li>Conditions <ul> <li>C1:L&lt;=0?</li> <li>C2: Objects are equal?</li> </ul> </li> <li>Actions <ul> <li>A1: Objects are equal</li> </ul> </li> <li>Table</li> </ul>					
	Test Case ID	C1	C2	Expected Output		
	TC1	Y/T	-	-		
	TC2	N/F	Y/T	True		
	TC3	N/F	N/F	False		
testGet	<ul> <li>Conditions <ul> <li>C1:L&lt;=0?</li> <li>C2: Index in range?</li> </ul> </li> <li>Actions <ul> <li>A1: Object is gotten</li> </ul> </li> <li>Table</li> </ul>					
	Test Case ID	C1	C2	Expected Output		
	TC1	Y/T	-	-		
	TC2	N/F	Y/T	Object at index		
	TC3	N/F	N/F	Error		
testIndexOf	<ul> <li>Conditions <ul> <li>C1:L&lt;=0?</li> <li>C2: Object in list?</li> </ul> </li> <li>Actions <ul> <li>A1: Index is gotten</li> </ul> </li> <li>Table</li> </ul>					
	Test Case ID	C1	C2	Expected Output		
	TC1	Y/T	-	-		
	TC2	N/F	Y/T	Index of object		
	TC3	N/F	N/F	-1		

testRemove	○ C: ○ C: • Actions	<ul> <li>C2: Object in list?</li> <li>Actions</li> <li>A1: Object is removed</li> </ul>					
	Test Case ID	C1	C2	Expected Output			
	TC1	Y/T	-	-			
	TC2	N/F	Y/T	Object removed			
	TC3	N/F	N/F	No change			
testRemoveAll	○ C: ○ C: • Actions	<ul> <li>C2: Object in list?</li> <li>Actions</li> <li>A1: Object is removed</li> </ul>					
	Test Case ID	C1	C2	Expected Output			
	TC1	Y/T	-	-			
	TC2	N/F	Y/T	Objects removed			
	TC3	N/F	N/F	No change			
testRetainAll	○ C2 ○ C2 • Actions	<ul> <li>C2: Objects exist to be retained?</li> <li>Actions</li> <li>A1: Objects retained</li> </ul>					
	Test Case ID	C1	C2	Expected Output			
	TC1	Y/T	-	-			
	TC2	N/F	Y/T	Objects retained			
	TC3	N/F	N/F	Empty List			
			1	<b>.</b>			

testSet	<ul> <li>Conditions <ul> <li>C1:L&lt;=0?</li> <li>C2: Index in range?</li> <li>C3: Object can be inserted</li> </ul> </li> <li>Actions <ul> <li>A1: Object at index is set</li> </ul> </li> <li>Table</li> </ul>					
	Test Case ID	C1	C2	C3	Expected Output	
	TC1	Y/T	1	-	-	
	TC2	N/F	Y/T	Y/T	Object at index is set	
	TC3	N/F	Y/T	N/F	Error	
	TC4	N/F	N/F	Y/T	Error	
	TC5	N/F	N/F	N/F	Error	
testSublist	○ C ○ C • Actions	1: Sub  C1  Y/T  N/F  N/F  N/F	t index index	in ranç		