KEY: Orange Highlight = Test added during this milestone

Red Highlight	= Impossible Test Case
---------------	------------------------

Test	Equivalence class partitioning(Wect, Sect, Wrect, Srect)
PredicatedListTest .testDecorated	<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></ul></li></ul>
PredicatedListTest .testEquals	<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> </ul> </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>
PredicatedListTest .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>
PredicatedListTest .testGet	<ul> <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 &lt; L</li> <li>I3: I &gt;= L</li> </ul> </li> <li>Wect</li> </ul>
	ID L I

	WECT1	L1	l1		
	WECT2	L2	12		
	WECT3	L1	13		
• Sect					
	ID	L	I		
	SECT1	L1	l1		
	SECT2	L2	I1		
	SECT3	L1	12		
	SECT4	L2	12		
	SECT5	L1	13		
	SECT6	L2	13		
<ul> <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> </ul>					
• Equival	Object O lence classes L1: L = 0 L2: L > 0 O1: O is in list				
	ID	L	0		
	WECT1	L1	01		
	WECT2	L2	O2		
• Sect					
	ID	L	I		
1					
	Wrect a SECT2     Wrect a SECT2     Wrect a SEQuival     Wect     Sect	WECT2 WECT3  Sect  ID SECT1 SECT2 SECT3 SECT4 SECT5 SECT6  Wrect and Srect would inve Wrect and srect are covere SECT2, SECT5, and SEC Wrect and srect are also or Variables:	WECT2 L2 WECT3 L1  • Sect  ID L SECT1 L1 SECT2 L2 SECT3 L1 SECT4 L2 SECT5 L1 SECT6 L2  • Wrect and Srect would involve L < 0 but this is every management of the second se		

		SECT2	L2	01
		SECT3	L1	O2
		SECT4	L2	O2
	<ul> <li>Wrect a</li> </ul>	and Srect would invented and srect are covered and srect are also countries.	ed by WECT1, SEC	T1
PredicatedListTest .tesetLastIndexOf	• Equiva •	es: List length L Object O lence classes L1: L = 0 L2: L > 0 O1: O is only instar O2: multiple instanc O3: O is not in list		
	Ü	ID	L	0
		WECT1	L1	O1
		WECT2	L2	O2
		WECT3	L1	О3
	• Sect			-
		ID	L	I
		SECT1	L1	01
		SECT2	L2	01
		SECT3	L1	O2
		SECT4	L2	O2
		SECT5	L1	O3
		SECT6	L2	О3
	<ul> <li>Wrect a SECT3</li> </ul>	and Srect would invented and srect are covered and srect are also cand srect are also cand	ed by WECT1, WE	CT3, SECT1,

PredicatedListTest .testRemove	• Equiva • • • • • • • • • • • • • • • • • • •	es: List length L Index I lence classes L1: L = 0 L2: L > 0 I1: I < 0 I2: I >= 0 && I < L I3: I >= L		
		ID	L	I
		WECT1	L1	I1
		WECT2	L2	12
		WECT3	L1	13
	● Sect ○			
	_	ID	L	I
		SECT1	L1	l1
		SECT2	L2	11
		SECT3	L1	12
		SECT4	L2	12
		SECT5	L1	13
		SECT6	L2	13
	<ul> <li>Wrect a SECT2</li> </ul>	and Srect would invented and srect are covered, SECT5, and SEC and srect are also county	ed by WECT1, WE0 T6	CT3, SECT1,
PredicatedListTest .testAdd	• Equiva • •	es: List length L Object O lence classes L1: L = 0 L2: L > 0 O1: O can be inser O2: O cannot be in		

			•		
		ID	L	0	
		WECT1	L1	O1	
		WECT2	L2	O2	
	• Sect				
		ID	L	1	
		SECT1	L1	O1	
		SECT2	L2	O1	
		SECT3	L1	O2	
		SECT4	L2	O2	
			olve L < 0 but this is ed by testing of a nu		
.testAddAll	<ul> <li>List length L</li> <li>List of objects O</li> <li>Equivalence classes</li> <li>L1: L = 0</li> <li>L2: L &gt; 0</li> <li>O1: O can be fully inserted into list</li> <li>O2: O can be partially inserted into list</li> <li>O3: O cannot be inserted at all into list</li> <li>Wect</li> </ul>				
		ID	L	0	
		WECT1	L1	O1	
		WECT2	L2	O2	
		WECT3	L1	O3	
	• Sect				
		ID	L	1	
		SECT1	L1	O1	
		SECT2	L2	01	
		SECT3	L1	O2	

		SECT4	L2		O2	
		SECT5	L1		O3	
		SECT6	L2		O3	
		and Srect would and srect are al			not possible a null variable list	
PredicatedListTest .testListIterator	<ul><li>Equival</li><li>O</li><li>Thus, wan none</li><li>O</li><li>Wrect a</li></ul>	List length L lence classes L1: L = 0 L2: L > 0 vect and sect a	done d involve L < 0	) but this is		
PredicatedListTest .testSet	<ul> <li>Variables:         <ul> <li>List length L</li> <li>Index I</li> <li>Object O</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 &lt; L</li> <li>I3: I &gt;= L</li> <li>O1: O can be inserted</li> <li>O2: O cannot be inserted</li> </ul> </li> <li>Wect</li> </ul>					
	0	ID	L	I	0	
		WECT1	L1	I1	01	
		WECT2	L2	12	O2	
		WECT3	L1	13	01	
	• Sect					
		ID	L	I	0	
		SECT1	L1	I1	01	
		SECT2	L2	l1	O1	

	SECT3	L1	12	O1
	SECT4	L2	12	01
	SECT5	L1	13	01
	SECT6	L2	13	01
	SECT7	L1	<b>I</b> 1	O2
	SECT8	L2	<b>I</b> 1	O2
	SECT9	L1	12	O2
	SECT10	L2	12	O2
	SECT11	L1	13	O2
	SECT12	L2	13	O2
0				

- Wrect and Srect would involve L < 0 but this is not possible</li>
   Wrect and srect are covered by WECT1, WECT3, SECT1, SECT2, SECT3, SECT5, SECT6, SECT7, SECT9, SECT10, SECT11, SECT12
- Wrect and srect are also covered by testing of a null variable list

Test	Boundary Valu	Boundary Value Analysis					
PredicatedListTest .testGet	<ul> <li>Variables:         <ul> <li>List length L</li> <li>Index I</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 &lt; I &lt; L, max-: I = L - 2, max: I = L - 1, max+: I = L}</li> </ul> </li> <li>BVA (basic)         <ul> <li>BVA (basic)</li> </ul> </li> </ul>						
		ID L I					
		BVA (basic) 1					
	BVA (basic) 2 L > 0 1						
	BVA (basic) 3 L > 0 nom						
		BVA (basic) 4	L > 0	L-2			

		BVA (basic) 5	L > (	0	L - 1
	BVA (ro	bust)			
		ID	L		I
		BVA (robust)	1 L>(	0	-1
		BVA (robust)	2 L>0	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	0	L - 1
		BVA (robust)	7 L>(	0	L
	•		•		
PredicatedListTest .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 &lt; I &lt; L, max-: I = L - 2, max: I = L - 1, max+: I = L}</li> <li>BVO: {nom: O can be inserted, nom: O cannot be inserted}</li> </ul> </li> <li>BVA (basic)</li> </ul>				
	0	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)	L > 0	L - 1	O can be

				T 1		
	5			inserted		
BVA (robust)						
0						
	ID	L	I	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		
	BVA (robust) 5	L > 0	L - 2	O can be inserted		
	BVA (robust) 6	L > 0	L - 1	O can be inserted		
	BVA (robust) 7	L > 0	L	O can be inserted		
 •						

Test	Decision Table					
PredicatedListTest .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 found to be equal</li> </ul> </li> <li>Table <ul> <li>Operations</li> <li>Table</li> </ul> </li> </ul>					
		Test Case ID	C1	C2	Expecte d Output	
		TC1	Y/T	-	-	
		TC2	N/F	Y/T	True	
		TC3	N/F	N/F	False	

Т

	0						
PredicatedListTest .testGet	<ul> <li>Conditions</li> <li>C1: L &lt;= 0?</li> <li>C2: Index in range?</li> <li>Actions</li> <li>A1: Object is gotten</li> <li>Table</li> </ul>						
	Č	Test Case ID	C1	C2	Expecte d Output		
		TC1	Y/T	-	-		
		TC2	N/F	Y/T	Object at index		
		TC3	N/F	N/F	Error		
	0						
.testIndexOf	<ul> <li>C1: L &lt;= 0?</li> <li>C2: Object in list?</li> <li>Actions</li> <li>A1: Index is gotten</li> <li>Table</li> </ul>						
		Test Case ID	C1	C2	Expecte d Output		
		TC1	Y/T	-	-		
		TC2	N/F	Y/T	Index of object		
		тС3	N/F	N/F	-1		
	0						
PredicatedListTest .tesetLastIndexOf	<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 <ul> <li></li> </ul> </li> </ul>						
		Test Case ID	C1	C2	Expecte d Output		

		TC1	Y/T	-	-		
		TC2	N/F	Y/T	Index of object		
		TC3	N/F	N/F	-1		
	0						
PredicatedListTest .testRemove	<ul> <li>Conditions <ul> <li>C1: L &lt;= 0?</li> <li>C2: Object in list?</li> </ul> </li> <li>Actions <ul> <li>A1: Object removed</li> </ul> </li> <li>Table <ul> <li>Object removed</li> </ul> </li> </ul>						
		Test Case ID	C1	C2	Expecte d Output		
		TC1	Y/T	-	-		
		TC2	N/F	Y/T	Object removed		
		тС3	N/F	N/F	No change		
PredicatedListTest .testAdd	<ul><li>Actions</li><li>Table</li></ul>	C1: L <= 0? C2: Object	can be inse	erted?			
	0	Test Case ID	C1	C2	Expecte d Output		
		TC1	Y/T	-	-		
		TC2	N/F	Y/T	Object inserted		
		TC3	N/F	N/F	No change		
	0						
PredicatedListTest .testAddAll	• Conditi	ons C1: L <= 0?	<b>&gt;</b>				

	<ul><li>Actions</li></ul>	C2: Objects	s can be i	nserted?			
		A1: Objects	are inse	rted			
	• Table					7	
		Test Case ID	C1	C2	Expecte d Output		
		TC1	Y/T	-	-		
		TC2	N/F	Y/T	Objects inserted		
		TC3	N/F	N/F	No change		
	0						
PredicatedListTest .testSet	<ul> <li>Conditions</li> <li>C1: L &lt;= 0?</li> <li>C2: Index in range?</li> <li>C3: Object can be inserted</li> <li>Actions</li> <li>A1: Object at index is set</li> </ul>						
	• Table o						
		Test Case ID	C1	C2	C3	Expecte d Output	
		TC1	Y/T	-	-	-	
		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	
	0						
PredicatedListTest .testSublist	<ul> <li>Conditions <ul> <li>C1: L &lt;= 0?</li> <li>C2: Start index in range?</li> <li>C3: End index in range?</li> </ul> </li> <li>Actions <ul> <li>A1: Sublist is made</li> </ul> </li> <li>Table <ul> <li><ul> <l><ul> <li><ul> <l><ul> <li><ul> <l><ul> <li><ul> <li>&lt;</li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></l></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></l></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></l></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul></li></ul>						

	Test Case ID	C1	C2	C3	Expecte d Output
	TC1	Y/T	-	-	-
	TC2	N/F	Y/T	Y/T	Sublist
	TC3	N/F	Y/T	N/F	Error
	TC4	N/F	N/F	Y/T	Error
	TC5	N/F	N/F	N/F	Error
0					

Test **Data Flow Coverage** PredicatedListTest .testEquals object Criterion Program Test Inputs Paths All-def <99, 100> <99, 100>  $(object = {2},$ 4, 6, 8, 10}) <99, 100> <99, 100>  $(object = {2},$ All-use 4, 6, 8, 10}) All-P-user-so <99, 100> <99, 100>  $(object = {2},$ 4, 6, 8, 10}) me-C-uses All-C-user-so <99, 100> <99, 100>  $(object = {2},$ 4, 6, 8, 10}) me-P-uses All DU paths <99, 100> <99, 100>  $(object = \{2,$ 4, 6, 8, 10}) PredicatedListTest .testGet Criterion index Program Test Inputs Paths All-def <110, 111> <110, 111> (index = 0)<110, 111> <110, 111> (index = 0)All-use All-P-user-so <110, 111> <110, 111> (index = 0)me-C-uses

		All-C-user-so me-P-uses	<110, 111>	<110, 111>	(index = 0)
		All DU paths	<110, 111>	<110, 111>	(index = 0)
PredicatedListTest .testIndexOf	•	Criterion	object	Program Paths	Test Inputs
		All-def	<115, 116>	<115, 116>	(object = 2)
		All-use	<115, 116>	<115, 116>	(object = 2)
		All-P-user-so me-C-uses	<115, 116>	<115, 116>	(object = 2)
		All-C-user-so me-P-uses	<115, 116>	<115, 116>	(object = 2)
		All DU paths	<115, 116>	<115, 116>	(object = 2)
PredicatedListTest .tesetLastIndexOf	•	Criterion	object	Program Paths	Test Inputs
		All-def	<120, 121>	<120, 121>	(object = 2)
		All-use	<120, 121>	<120, 121>	(object = 2)
		All-P-user-so me-C-uses	<120, 121>	<120, 121>	(object = 2)
		All-C-user-so me-P-uses	<120, 121>	<120, 121>	(object = 2)
		All DU paths	<120, 121>	<120, 121>	(object = 2)
PredicatedListTest .testRemove	•	Criterion	index	Program Paths	Test Inputs
		All-def	<125, 126>	<125, 126>	(object = 2)
		All-use	<125, 126>	<125, 126>	(object = 2)

	All-P-use me-C-use		0	<125, 12	6>	<125, 126>		(object = 2)	
		All-C-user-so me-P-uses		<125, 12	26> <12		25, 126>		bject = 2)
		All DU paths	3	<125, 12	6>	<125	, 126>	(0	bject = 2)
PredicatedListTest	•								
.testAdd		Criterion	ind	dex	object		Program Paths		Test Inputs
		All-def		31, 33>	<131, 132>		<131, 13 133>	2,	(object = 2)
		All-use	<131, 133>		<131, 132> <131, 132, 133>		<131, 13 133>	2,	(object = 2)
			<131, 133>		<131, 132>		<131, 13 133>	2,	(object = 2)
		All-C-user- some-P-us es		31, 33>	<131, 132> <131, 133>		<131, 13 133>	2,	(object = 2)
		All DU paths		31, 33>	<131, 132> <131, 133>		<131, 13 133>	2,	(object = 2)
PredicatedListTest	•	<u> </u>							
.testAddAll		Criterion	ind	dex	coll		Program Paths		Test Inputs
		All-def	{1: 13	37, 38, 9}*, 88, 141>	<137, 138>		<137, {138, 139}*, 138, 141	>	(coll = {12, 14, 16})
		All-use	{1	37, 38, 9}*,	<137, {138, 139}*,		<137, {138, 139}*,		(coll = {12, 14, 16})

		138, 141>	138> <137, {138, 139}*, 138, 139> <137, {138, 139}*, 138, 139, 141>	138, 141>	
	All-P-uses -some-C-u ses	<137, {138, 139}*, 138, 141>	<137, {138, 139}*, 138>	<137, {138, 139}*, 138, 141>	(coll = {12, 14, 16})
	All-C-user- some-P-us es	<137, {138, 139}*, 138, 141>	<137, {138, 139}*, 138> <137, {138, 139}*, 138, 139> <137, {138, 139}*, 139, 141>	<137, {138, 139}*, 138, 141>	(coll = {12, 14, 16})
	All DU paths	<137, {138, 139}*, 138, 141>	<137, {138, 139}*, 138> <137, {138, 139}*, 138, 139> <137, {138, 139}*, 138, 139, 141>	<137, {138, 139}*, 138, 141>	(coll = {12, 14, 16})
PredicatedListTest	•				
.testListIterator	Criterion			Progran	m Paths

		All-def				
		All-use				
		All-P-user-s	ome-C-			
		All-C-user-s uses	ome-P-			
		All DU paths	5			
PredicatedListTest	•		1		Ī	1
.testSet		Criterion	index	object	Program	Test Inputs

				i
Criterion	index	object	Program Paths	Test Inputs
All-def	<155, 156, 157>	<155, 156>	<155, 156, 157>	(index = 0, object = 0)
All-use	<155, 156, 157>	<155, 156> <155, 156, 157>	<155, 156, 157>	(index = 0, object = 0)
All-P-user- some-C-u ses	<155, 156, 157>	<155, 156>	<155, 156, 157>	(index = 0, object = 0)
All-C-user- some-P-us es	<155, 156, 157>	<155, 156> <155, 156, 157>	<155, 156, 157>	(index = 0, object = 0)
All DU paths	<155, 156, 157>	<155, 156> <155, 156, 157>	<155, 156, 157>	(index = 0, object = 0)