

KEY: Orange highlight = Test Added during this milestone

Red highlight = Impossible test case

Test	Equivalence class partitioning(Wect, Sect, Wrect, Srect)																								
testAddNodeToCache	<ul style="list-style-type: none">Variables:<ul style="list-style-type: none">List remaining cache sizeObject OEquivalence classes<ul style="list-style-type: none">L1: L=0L2: L>0O1: O does not already exist in listO2: O already exists in listWect<table><tr><td>ID</td><td>L</td><td>O</td></tr><tr><td>WECT1</td><td>L1</td><td>O1</td></tr><tr><td>WECT2</td><td>L2</td><td>O2</td></tr></table>Sect<table><tr><td>ID</td><td>L</td><td>I</td></tr><tr><td>SECT1</td><td>L1</td><td>O1</td></tr><tr><td>SECT2</td><td>L2</td><td>O1</td></tr><tr><td>SECT3</td><td>L1</td><td>O2</td></tr><tr><td>SECT4</td><td>L2</td><td>O2</td></tr></table>Wrect and Srect would involve L < 0 but this is not possibleWrect and srect are covered by testing of a null variable list	ID	L	O	WECT1	L1	O1	WECT2	L2	O2	ID	L	I	SECT1	L1	O1	SECT2	L2	O1	SECT3	L1	O2	SECT4	L2	O2
ID	L	O																							
WECT1	L1	O1																							
WECT2	L2	O2																							
ID	L	I																							
SECT1	L1	O1																							
SECT2	L2	O1																							
SECT3	L1	O2																							
SECT4	L2	O2																							
testGetMaximumCacheSize	<ul style="list-style-type: none">Variables:<ul style="list-style-type: none">Cache maximum sizeEquivalence classes<ul style="list-style-type: none">L1: L=0L2: L>0Thus, wect and sect are both covered by using an empty list and a nonempty<ul style="list-style-type: none">This is already doneWrect and Srect would involve L < 0 but this is not possibleWrect and rect are covered by testing of a null variable list																								
testShrinkCacheTo	<ul style="list-style-type: none">Variables:																								

MaximumSize	<ul style="list-style-type: none"><ul style="list-style-type: none">List cache LMaximum cache size MEquivalence classes<ul style="list-style-type: none">L1: L=0L2: L>0M1: M<0M2: M>=0 && M<LM3: M >= LWect<table><tr><td>ID</td><td>L</td><td>M</td></tr><tr><td>WECT1</td><td>L1</td><td>M1</td></tr><tr><td>WECT2</td><td>L2</td><td>M2</td></tr><tr><td>WECT3</td><td>L1</td><td>M3</td></tr></table>Sect<table><tr><td>ID</td><td>L</td><td>M</td></tr><tr><td>SECT1</td><td>L1</td><td>M1</td></tr><tr><td>SECT2</td><td>L2</td><td>M1</td></tr><tr><td>SECT3</td><td>L1</td><td>M2</td></tr><tr><td>SECT4</td><td>L2</td><td>M2</td></tr><tr><td>SECT5</td><td>L1</td><td>M3</td></tr><tr><td>SECT6</td><td>L2</td><td>M3</td></tr></table>Wrect and Srect would involve L < 0 but this is not possibleWrect and srect are also covered by testing of a null variable list	ID	L	M	WECT1	L1	M1	WECT2	L2	M2	WECT3	L1	M3	ID	L	M	SECT1	L1	M1	SECT2	L2	M1	SECT3	L1	M2	SECT4	L2	M2	SECT5	L1	M3	SECT6	L2	M3
ID	L	M																																
WECT1	L1	M1																																
WECT2	L2	M2																																
WECT3	L1	M3																																
ID	L	M																																
SECT1	L1	M1																																
SECT2	L2	M1																																
SECT3	L1	M2																																
SECT4	L2	M2																																
SECT5	L1	M3																																
SECT6	L2	M3																																
testGetNodeFrom Cache	<ul style="list-style-type: none">Variables:<ul style="list-style-type: none">List cache LIndex IEquivalence classes<ul style="list-style-type: none">L1: L=0L2: L>0I1: I<0I2: I>=0 && I<LI3: I>=LWect																																	

	<table><tr><td>ID</td><td>L</td><td>I</td></tr><tr><td>WECT1</td><td>L1</td><td>I1</td></tr><tr><td>WECT2</td><td>L2</td><td>I2</td></tr><tr><td>WECT3</td><td>L1</td><td>I3</td></tr></table> <ul style="list-style-type: none">• Sect<table><tr><td>ID</td><td>L</td><td>I</td></tr><tr><td>SECT1</td><td>L1</td><td>I1</td></tr><tr><td>SECT2</td><td>L2</td><td>I1</td></tr><tr><td>SECT3</td><td>L1</td><td>I2</td></tr><tr><td>SECT4</td><td>L2</td><td>I2</td></tr><tr><td>SECT5</td><td>L1</td><td>I3</td></tr><tr><td>SECT6</td><td>L2</td><td>I3</td></tr></table>• Wrect and Srect would involve $L < 0$ but this is not possible• Wrect and srect are also covered by testing of a null variable list	ID	L	I	WECT1	L1	I1	WECT2	L2	I2	WECT3	L1	I3	ID	L	I	SECT1	L1	I1	SECT2	L2	I1	SECT3	L1	I2	SECT4	L2	I2	SECT5	L1	I3	SECT6	L2	I3
ID	L	I																																
WECT1	L1	I1																																
WECT2	L2	I2																																
WECT3	L1	I3																																
ID	L	I																																
SECT1	L1	I1																																
SECT2	L2	I1																																
SECT3	L1	I2																																
SECT4	L2	I2																																
SECT5	L1	I3																																
SECT6	L2	I3																																
testIsCacheFull	<ul style="list-style-type: none">• Variables:<ul style="list-style-type: none">◦ Remaining cache size R• Equivalence classes<ul style="list-style-type: none">◦ R1: $R=0$◦ R2: $R>0$• Thus, wect and sect are both covered by using an empty list and a nonempty<ul style="list-style-type: none">◦ This is already done• Wrect and Srect would involve $R < 0$ but this is not possible• Wrect and rect are covered by testing of a null variable list																																	
testNodeCachingLinkedList	<ul style="list-style-type: none">• Variables:<ul style="list-style-type: none">◦ List Cache L• Equivalence classes<ul style="list-style-type: none">◦ L1: $L=0$◦ L2: $L>0$• Thus, wect and sect are both covered by using an empty list and a nonempty<ul style="list-style-type: none">◦ This is already done• Wrect and Srect would involve $L < 0$ but this is not possible• Wrect and rect are covered by testing of a null variable list																																	

Test	Boundary Value Analysis																																										
testGetNodeFromCache	<ul style="list-style-type: none">Variables:<ul style="list-style-type: none">List cach LIndex ITest Set<ul style="list-style-type: none">BVL: {nom:L=0,nom:L>0}BVI: {min-:I=-1,min:I=0,min+:I=1,nom:1<I<L,max-: I = L - 2, max: I = L - 1, max+: I = L}BVA (basic)<table><tr><th>ID</th><th>L</th><th>I</th></tr><tr><td>BVA (basic) 1</td><td>L>0</td><td>0</td></tr><tr><td>BVA (basic) 2</td><td>L>0</td><td>1</td></tr><tr><td>BVA (basic) 3</td><td>L>0</td><td>nom</td></tr><tr><td>BVA (basic) 4</td><td>L>0</td><td>L-2</td></tr><tr><td>BVA (basic) 5</td><td>L > 0</td><td>L - 1</td></tr></table>BVA (robust)<table><tr><th>ID</th><th>L</th><th>I</th></tr><tr><td>BVA (robust) 1</td><td>L>0</td><td>-1</td></tr><tr><td>BVA (robust) 2</td><td>L>0</td><td>0</td></tr><tr><td>BVA (robust) 3</td><td>L>0</td><td>1</td></tr><tr><td>BVA (robust) 4</td><td>L>0</td><td>nom</td></tr><tr><td>BVA (robust) 5</td><td>L>0</td><td>L-2</td></tr><tr><td>BVA (robust) 6</td><td>L>0</td><td>L-1</td></tr><tr><td>BVA (robust) 7</td><td>L>0</td><td>L</td></tr></table>	ID	L	I	BVA (basic) 1	L>0	0	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	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
ID	L	I																																									
BVA (basic) 1	L>0	0																																									
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																																									
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																
testAddNodeToCache	<ul style="list-style-type: none">Conditions<ul style="list-style-type: none">C1: L<=0?C2: Object can be inserted?Actions<ul style="list-style-type: none">A1: Object is insertedTable<table><tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr><tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr><tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object inserted</td></tr><tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr></table>	Test Case ID	C1	C2	Expected Output	TC1	Y/T	-	-	TC2	N/F	Y/T	Object inserted	TC3	N/F	N/F	No change
Test Case ID	C1	C2	Expected Output														
TC1	Y/T	-	-														
TC2	N/F	Y/T	Object inserted														
TC3	N/F	N/F	No change														
testGetMaximumCacheSize	<ul style="list-style-type: none">Conditions<ul style="list-style-type: none">C1: L<=0?C2: max size in range?Actions<ul style="list-style-type: none">A1: Size is gottenTable<table><tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr><tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr><tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object at index</td></tr><tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Error</td></tr></table>	Test Case ID	C1	C2	Expected Output	TC1	Y/T	-	-	TC2	N/F	Y/T	Object at index	TC3	N/F	N/F	Error
Test Case ID	C1	C2	Expected Output														
TC1	Y/T	-	-														
TC2	N/F	Y/T	Object at index														
TC3	N/F	N/F	Error														
testShrinkCacheToMaximumSize	<ul style="list-style-type: none">Conditions<ul style="list-style-type: none">C1: L<=0?C2: Current size in range?Actions<ul style="list-style-type: none">A1: Cache size shrinksTable<table><tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr><tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr><tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object at index</td></tr><tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Error</td></tr></table>	Test Case ID	C1	C2	Expected Output	TC1	Y/T	-	-	TC2	N/F	Y/T	Object at index	TC3	N/F	N/F	Error
Test Case ID	C1	C2	Expected Output														
TC1	Y/T	-	-														
TC2	N/F	Y/T	Object at index														
TC3	N/F	N/F	Error														

testGetNodeFrom Cache	<ul style="list-style-type: none">● Conditions<ul style="list-style-type: none">○ C1:L<=0?○ C2: Index in range?● Actions<ul style="list-style-type: none">○ A1: Object is gotten● Table<table><tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr><tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr><tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object at index</td></tr><tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Error</td></tr></table>	Test Case ID	C1	C2	Expected Output	TC1	Y/T	-	-	TC2	N/F	Y/T	Object at index	TC3	N/F	N/F	Error
Test Case ID	C1	C2	Expected Output														
TC1	Y/T	-	-														
TC2	N/F	Y/T	Object at index														
TC3	N/F	N/F	Error														
testIsCacheFull	<ul style="list-style-type: none">● Conditions<ul style="list-style-type: none">○ C1:L<=0?○ C2: Cache is full?● Actions<ul style="list-style-type: none">○ A1: Check cache size and max size● Table<table><tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr><tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr><tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>True</td></tr><tr><td>TC3</td><td>N/F</td><td>N/F</td><td>False</td></tr></table>	Test Case ID	C1	C2	Expected Output	TC1	Y/T	-	-	TC2	N/F	Y/T	True	TC3	N/F	N/F	False
Test Case ID	C1	C2	Expected Output														
TC1	Y/T	-	-														
TC2	N/F	Y/T	True														
TC3	N/F	N/F	False														