Write out your test cases in this document before implementing them in code. Each function requires at least two tests, but possibly more to cover reasonable usage.

Reference the **Function Specifications** section in the main project specification for information on the way the functions work.

Turn in this sheet under the project assignment on Canvas.

bool IsEmpty() const

Test #	State setup	Inputs	Expected output
1.	Create a List, do nothing	(none)	IsEmpty() returns true
2.	Create a List, insert 1 item	(none)	IsEmpty() returns false

We should make sure IsEmpty() works for both the true and false cases.

bool IsFull() const

Test #	State setup	Inputs	Expected output
1.	Create a list, do nothing	(none)	IsFull() returns false
2.	Create a list, insert 5 items	(none)	IsFull() returns false
3.	Create a list, insert 100 items	(none)	IsFull() returns true
4.	Create a list, try to insert 101 items	(none)	IsFull() returns true.

Here I’m testing for 5 items just to make sure something doesn’t go wrong with the inserts that triggers it to be flagged as “full”. I’m also checking if we try to add *over* the max amount that it still says “full” to make sure that the code to insert a new item doesn’t mess up the item count if the insert fails.

int Size() const

Test #	State setup	Inputs	Expected output
1.	Create a List. Do nothing.	(none)	Size() returns 0
2.	Create a List. Insert 5 items.	(none)	Size() returns 5
3.	Create a List. Insert 101 items.	(none)	Size() returns 100

Here we are also making sure that if an insert fails (the 101st item), it doesn't incorrectly increment the size when it's already full.

int GetCountOf(const T& item) const

Test #	State setup	Inputs	Expected output
1.	Create a List. Do nothing.	"A"	GetCountOf() returns 0
2.	Create a List. Insert "A", "B", "C"	"A"	GetCountOf() returns 1
3.	Create a List. Insert "C", "A", "K", "E", "A", "A"	"A", "Z"	GetCountOf("A") returns 3 GetCountOf("Z") returns 0

Here we want to make sure that it counts all instances of "A", whether it's at the beginning, end, or middle of the list.

bool Contains(const T& item) const

Test #	State setup	Inputs	Expected output
1.	Create a List. Do nothing.	"A"	Contains() returns false
2.	Create a List. Insert "A", "B", "C"	"C"	Contains() returns true

List() Constructor

Test #	State setup	Inputs	Expected output
1.	Create a List	(none)	Size() returns 0.

Making sure the constructor sets up the List in the correct state.

bool PushFront(const T& newItem)

Test #	State setup	Inputs	Expected output
1.	Create a List.	“A”	PushFront(“A”) returns true. Get(0) returns “A” Size() returns 1
2.	Create a List.	“A”, “B”, “C”	PushFront(“A”), PushFront(“B”), and PushFront(“C”) all return true. Get(0) returns “C” Get(1) returns “B” Get(2) returns “A” Size() returns 3
3.	Create a List. Insert 100 items.	Insert the 101th item, “A”	PushFront(“A”) returns false. Size() returns 100

Here we are making sure that if a Push is successful, we get true, and false if not. Also we are verifying that the items are added in the correct order.

bool PushBack(const T& newItem)

Test #	State setup	Inputs	Expected output
1.	Create a List.	“A”	PushBack(“A”) returns true. Get(0) returns “A” Size() returns 1
2.	Create a List.	“A”, “B”, “C”	PushBack(“A”), PushBack(“B”), and PushBack(“C”) all return true. Get(0) returns “A” Get(1) returns “B” Get(2) returns “C” Size() returns 3
3.	Create a List. Insert 100 items.	Insert the 101th item, “A”	PushBack(“A”) returns false. Size() returns 100

Again we’re testing to make sure the items are added in the correct order, and the return values work.

bool Insert(int atIndex, const T& item)

Test #	State setup	Inputs	Expected output
1.	Create a List.	Insert(0, “A”)	Insert(0, “A”) returns true Size() returns 1 Get(0) returns “A”
2.	Create a List.	Insert(50, “A”)	Insert(50, “A”) returns false (not contiguous) Size() returns 0
3.	Create a List	Insert(0, “A”) Insert(1, “B”) Insert(1, “C”)	The inserts return true. Get(0) returns “A” Get(1) returns “B” Get(2) returns “C” Size() returns 3

T* Get(int atIndex)

Test #	State setup	Inputs	Expected output
1.	Create a List. Run PushBack(“A”), PushBack(“B”), and PushBack(“C”)	0 1 2	Get(0) returns “A” Get(1) returns “B” Get(2) returns “C”
2.	Create a List.	0	Get(0) returns nullptr

T* GetFront()

Test #	State setup	Inputs	Expected output
1.	Create a List.	(none)	GetFront() returns nullptr
2.	Create a List. Push back “A”, “B”, “C”	(none)	GetFront() returns “A”

T* GetBack()

Test #	State setup	Inputs	Expected output
1.	Create a List.	(none)	GetBack() returns nullptr
2.	Create a List. Push back “A”, “B”, “C”	(none)	GetBack() returns “C”

bool PopFront()

Test #	State setup	Inputs	Expected output
1.	Create a List.	(none)	PopFront() returns false
2.	Create a List. PushBack “A”, “B”, “C”	(none)	PopFront() returns true GetFront() returns “B”

Making sure Pop removes the correct item.

bool PopBack()

Test #	State setup	Inputs	Expected output
1.	Create a List.	(none)	PopBack() returns false
2.	Create a List. PushBack “A”, “B”, “C”	(none)	PopBack() returns true GetBack() returns “B”

bool Remove(const T& item)

Test #	State setup	Inputs	Expected output
1.	Create a List. Push back “A”, “B”, “C”	Remove(“B”)	Remove(“B”) returns true Get(0) returns “A” Get(1) returns “C” Size() returns 2
2.	Create a List. Push back “A”, “B”, “C”	Remove(“Z”)	Remove(“Z”) returns false Size() returns 3
3.	Create a List. Push back “C”, “H”, “E”, “E”, “S”, “E”	Remove (“E”)	Get(0) returns “C” Get(1) returns “H” Get(2) returns “S” Size() returns 3

bool Remove(int atIndex)

Test #	State setup	Inputs	Expected output
1.	Create a List. Push back “A”, “B”, “C”	Remove(0)	Remove(0) returns true Get(0) returns “B” Get(1) returns “C” Size() returns 2
2.	Create a List. Push back “A”, “B”, “C”	Remove(5)	Remove(5) returns false Size() returns 3
3.	Create a List. Push back “C”, “H”, “E”, “E”, “S”, “E”	Remove (2)	Get(0) returns “C” Get(1) returns “H” Get(2) returns “E” Get(3) returns “S” Get(4) returns “E” Size() returns 5

void Clear()

Test #	State setup	Inputs	Expected output
1.	Create a List. Push back “A”, “B”, “C”	(none)	Size() returns 0 IsEmpty() returns true

bool ShiftRight(int atIndex)

Test #	State setup	Inputs	Expected output
1.	Create a List. Push Back “A”, “B”, “C”	ShiftRight(1)	Get(0) returns “A” Get(2) returns “B” Get(3) returns “C” We don’t care what’s at position 1.

bool ShiftLeft(int atIndex)

Test #	State setup	Inputs	Expected output
1.	Create a List. Push Back “A”, “B”, “C”	ShiftLeft(1)	Get(0) returns “A” Get(1) returns “C” We don’t care what’s at position 2.