Phase A: Direct-Mapped Memory			
Test Case #	Requirement	Test Description and Input Data	Expected result/output
1	Test an invalid input to malloc	malloc(0)	NULL
2	Test valid input to malloc	malloc(sizeof(int)) assign data to it	Some pointer
3	Test an invalid input to free	free(0)	Nothing happens
4	Test an invalid input to free	X = malloc(4) Free(x+2)	Nothing happens
5	Test a valid input to free	malloc(4) free it	Nothing happens
6	Double free	malloc(4) and free it twice	Nothing happens
7	Malloc very large size	malloc(PAGE_SIZE+4) assign data to whole size	Some pointer
8	Malloc in order, free in order	malloc three ints, A, B, C, assign data to them, then free A, B, C	Three pointers, three frees
9	Malloc in order, free out of order	malloc A, B, C, assign data, free B, C, A	Three pointers, three frees

Phase B: Virtual(ish) Memory			
Test Case #	Requirement	Test Description and Input Data	Expected result/output
1	Test two pointer positions	Create two threads, have them malloc an int, and ensure that the int has the same pointer but different values in both threads	Pointers are the same
2	Ensure data is placed back where it should go	Create two threads, malloc more than a page length in both threads, fill with data, and test to be sure that data is still there	Data is there
3	Virtualization handles multiple page allocations across threads	Malloc X amount of pages on one thread, make a variable on one of the pages it sleep(1). Have another thread allocate X amount of pages and allocate a variable on one of them. Have them try to change and access the variables successfully	Variable change is reflected; Pointers for threads are same
4	Malloc uses the first free page		

Phase C: Swap File			
Test Case #	Requirement	Test Description and Input Data	Expected result/output
1	Max out pages; Show that swapping works even with heap and file full	Using one or more threads, allocate until null. Note the number of pages.	The maximum number of pages available.
2	Can malloc after filling the heap	Fill up the heap with more than pages than it can hold. Malloc a variable and access it. Show a valid pointer.	Pointer is not null
2	Data change after filling pages	Fill up the heap with more pages than it can hold. Malloc a variable and change alter its value.	Data change is reflected

Phase D: Shalloc			
Test Case #	Requirement	Test Description and Input Data	Expected result/output
1	Bound checking for shalloc	Try to allocate more than 4 pages worth of memory; try to allocate less than a size of 1	Pointer should be NULL
2	Pointers are different for multiple shallocs across threads	Have threads shalloc variables. Their pointers should all be an offset from each other within the shalloc page	Pointers are different
3	Data change is shared	Shalloc a global variable and change it in a thread. Have another thread access that variable.	Values are the same
4	Data change is reflected from free	Shalloc a variable in main and alter its value. Have another thread free that shalloc variable. Have another thread show that the pointer is NULL from the free.	Pointer is NULL