- 60% penalty if you use STL/external data structures for book keeping since you cannot guarantee their memory footprint (have your simple data structures like hashmap, linked list etc. where you have absolute control over the overhead for these data structures)

- Use demo1.c and demo2.c (and any other demo code you want to write) to show working of grading parts below

- 90% penalty if your code does not print meaningful debug messages while its working in each step (data structure creation/destruction/updation, computation, invoking functions)

- Many of the partial marks in different parts are already part of report (e.g., locks, design considerations). So just point to the report while explaining your logic for those parts.

Creation of memory (createMem) 10 marks	• mallocing the correct amount of data • separating out the book keeping space • Design considerations for efficiency (in terms of memory footprint / speed) • Printing message	1 4 4 1
Creation/destruction of variable/array (createVar/CreateArr) 15 marks	• Correctly creating/destroying variable/array • Implementing data types (int, char, medium int, boolean) • Finding valid free segment with a rational algorithm • Designing and Implementing special data structures • Freeing up memory algorithm and data structure implementation • Error handling • Design considerations for efficiency (in terms of memory footprint / speed) • Printing message	2 1 2 3 3 1 2 1
Assigning value to a variable/Array (assignvar/assignArr)	• Correctly assigning values • Handling data types (int, char, medium int, boolean)	2 2

10 marks	• type checking • Design considerations for efficiency (in terms of handling data type and type checking) • Printing message	2 3 1
freeElem 10 marks	• Correctly "free"-ing memory used by variables • Calling freeElem in appropriate place and rationalizing decision • Design considerations for efficiency (in terms of speed/memory penalty, e.g., freeing in batch etc.) • Printing message	3 3 3 1
Word alignment 10 marks	• Ensuring interaction with the assigned memory (created by createMem function) must access data chunks of 4 bytes • Tracking the actual size of the variable • Design considerations for efficiency (in terms of speed/memory, e.g., reusing holes etc.) • Printing message	4 2 3 1
Garbage collection 25 marks	• Correctly freeing up appropriate memory after function invoked. • Running garbage collector (GC) in thread • Invocation of GC in correct places • Design / Implementation of algorithm for garbage collector • Design / implementation of data strutures for GC • Design / Implementation of using locks for appropriate data structures • Design / Implementation of algorithm for compaction • Design / implementation of data structures for compaction • Invocation of compact function in correct places	2 2 2 4 3 2 3 3 1

	• Design considerations for efficiency (in terms of speed/memory penalty) • Printing message	2 1
Demo codes with library 10 marks	• demo1.c runs as expected • demo2.c runs as expected • Using extra demo and explanation to show working of your special considerations	4 4 2
Report 10 marks	• Structure of local page table (or symbol table) with justification (what does each element do) • Additional data structure list • Algo of compact function and logic of invoking it • Impact of GC in terms of memory footprint • Rationalizing usage of locks in the library	2 2 1 4 1