(1) Workload Distribution
Part A Humeyra, Gozde, Irem
Part B Humeyra, Gozde, Irem
Part C Humeyra, Gozde, Irem and
Report part all of us contribute on it.

(2) All parts worked properly, and we checked with test cases which are given.

(3) Approach to implementations

For stack and queue push operations restricted given in Assumptions and Constraints part in project description hence we book for 1000 elements. In queue implementation we book 1003, index 0 is for size of queue, index 1001 for header of queue, and index 1002 for tail of the queue. When we push a new item to the queue the length of the queue increases so we increased the value of index 0 by 1, and increased value of index 1002 (tail) since we are adding the new item at the end of the queue. When we pop the queue, the length is decreased so we decrease the value of index 0 by 1. New header is getting the next item hence we shift in way of increase the index 1001 by 1. Because of deletion, first element is not the header of the queue anymore hence we set bool #f, and we return pop-val. In case of top operation, the length of the queue does not change, and the first element is shown. Hence, we did not change the header (index 1001) since there is not any deletion and return first element with top-val.

In stack implementation when we create new stack, we book 1001 in newstack-exp( ) index 0 is for length of stack and #f since the stack is empty. After creation of new stack, the stack is empty so we initialize the length zero at the beginning. In push operation, when we add new item length also increase hence we increase value of index 0 by 1 (which gives us length of stack), and add the item at the end of stack. In stack-pop-exp (exp1) we take the first element of the stack, which is last added to the stack, and we decrease the counter by 1 because of deletion and return the top item with pop-val at the end of running. In stack-top-exp method we check the stack-array is empty or not. If it is empty, we return -1 (num-val -1) which is given us in project description. We did not change counter since there is not any deletion and return last added element with top-val. In lang.scm folder we describe expressions of stack with linkable modules of our language with newstack, stack-push, stack-pop, stack size, stack-top, empty-stack?, print-

stack. In array implementation we define array in make-array method which uses memory allocation in scheme for count and header. In array-set! method, we check array via array-chk then setref! the header to update-array-exp, besides we use stack and queue implementations which is based on array, in our language.