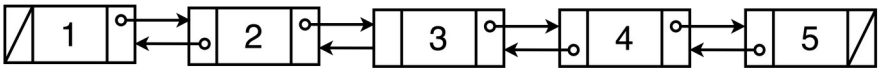
Q1. Write **ADT code** for an *insert* function for the *list* ADT in a way that the values are inserted in the sorted position in the list. For example, after inserting the values 4, 1, 5, 3, 2 in the list, the list should look like as shown in the following figure. The code should be generic and should also work for any other values. [10]



Q2. Write **ADT code** for a function *delete_smaller* for the linked structures based queue ADT. The function should take a number as a parameter and delete all the values smaller than that number. For example, if the queue contains the values 4, 1, 5, 3, 2, after calling the function *delete_smaller(3)*, the queue would contain the values 4, 5, 3. The time complexity of the function should not exceed O(n). The code should be generic and should also work for any other values. [10]

Q3. Write **ADT code** of *reverse_iterator* for the *list* ADT (doubly linked list). Only write code for the iterator and its functions, not for the *list* ADT. [10]

Q4. Write **ADT code** for copy constructor for linked structures based stack ADT. [5]

Q5. Write non-recursive/iterator **ADT code** for the *ReheapUp* function of the min-heap data structure. [5]

Q6. A *zigzag* traversal in a binary search tree starts from the root node, then visits its right child, then the left child of the right child and so on. For example, in the following figure, calling the *zigzag* function should display the highlighted values 4, 8, 5, 7, 6. Write **ADT code** for the *zigzag* function. The code should be generic and should also work for any other values and tree of any height. [10]