NAME: Goutami Padmanabhan ID:1001669338

Task 1

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test case | Data/code | Does my code handle it?/Time Complexity |
| sublist**(list A, list pos\_list)** | Index out of bounds | A: 10 ->10 ->40 ->20  pos\_list: (**-7**) -> 3 or  pos\_list: 3 -> **80000** -> 3  result: fct returns NULL | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A is NULL | list A = NULL;  result: fct returns NULL | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A is empty | list A = newList();  result: fct returns NULL | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | pos\_list is empty | list pos\_list = NULL;  result: fct returns NULL | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | pos\_list is NULL | list pos\_list = newList(); result: fct returns NULL | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A is not modified by sublist(…)  …. | A: 15 -> 100 -> 7 -> 5 -> 100  pos\_list: 3 -> 0 ->2  result: A will still be :  15 -> 100 -> 7 -> 5 -> 100 | Yes  O(N)  It traverses the entire list |
|  | Normal data  (as in hw writeup) | A: 15 -> 100 -> 7 -> 5 -> 100 -> 7 -> 30  pos\_list: 3 -> 0 -> 6 -> 4 | 5->15->30->100  O(N)  It traverses the entire list |
|  | Repeated position | A: 5  pos\_list: 0 -> 0 -> 0  result: returns: 5-> 5-> 5 | Yes  O(N)  It traverses the entire list |
|  |  |  |  |
| **deleteOccurrences**  **(list A, int V)** | Normal data, V is in A  (as in hw write-up) | A: 15 -> 100 -> 7 -> 5 -> 100 -> 7 -> 30  V is 7,  Result: A will become:  15-> 100-> 5 -> 100 -> 30 | Yes  O(N)  It traverses the entire list |
|  | V does not occur in A | A: 15 -> 100 -> 7 -> 5  V is 9,  Result: A does not change:  15-> 100-> 7-> 5 | Yes  O(N)  It traverses the entire list |
|  | Repeated consecutive occurrences | A: 15 -> 7 -> 7 -> 5  V is 7,  Result: A becomes:  15 -> 5 | Yes  O(N)  It traverses the entire list |
|  | A has one item and that is V | A: 7  V is 7  Result: A becomes Empty | Yes  O(N) or O(1)  It traverses the entire list |
|  | A has only items with value V in it | A: 7->7-> 7  V is 7  Result: A becomes empty | Yes  O(N)  It traverses the entire list |
|  | A is NULL | A = NULL  Result: A is not changed | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A is empty | A = newList()  Result: A is not changed | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  |  |  |  |
| **swapFirstThird (list A)** | A is NULL | A = NULL  Result: A is not changed | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A is empty | A = newList()  Result: A is not changed | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A has one item | A: 7  Result: A does not change | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A has two items | A: 7->10  Result: A becomes:  10->7 | Yes  O(1)  Because the list is smaller than three elements, it’s going to be a constant time |
|  | A has three or more items | A: 15 -> 100 -> 7 -> 5 -> 100 -> 7 -> 30  Result: A becomes:  7->100->15->5->100->7->30 | Yes  O(1)  Since we have to traverse only till the third element each time and change the pointers. N does not matter as we do not have to traverse through the entire list |
|  |  |  |  |
| **moveAllMaxAtEnd**  **(list A)** | A is NULL | A = NULL  Result: A is not changed | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A is empty | A = newList()  Result: A is not changed | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | Normal data  (as in hw write-up) | A: 15 -> 100 -> 5 -> 100 -> 30  Result: A will become:  15 -> 5 -> 30 -> 100 -> 100 | Yes  O(N)  It traverses the entire list |
|  | A has one item | A: 7  Result: A does not change | Yes  O(1)  It runs in constant time. It doesn’t depend on N. It doesn’t traverse the list |
|  | A has only items of the same value in it (all items are MAX). | A: 7-> 7 ->7  Result: A does not change (the order of the nodes does not change either) | Yes  O(N)  It traverses the entire list |
|  | MAX is on first position | A: 100-> 7->20  Result: A: 7->20->100 | Yes  O(N)  It traverses the entire list |
|  | MAX is on last position | A: 10-> 7->200  Result: A: 10->7->200 | Yes  O(N)  It traverses the entire list |

**CODE & DRAWING for swapFirstThird (list A) (This is a reminder of what needs to be done. Do not squeeze the answer in here. Use an additional page.)**

Task 2 :

The time complexity for the functions written has been specifies in the tabular column already present above. Θ is represented as O since I had issues while converting word document to pdf

Task 3(10 points)Given:

|  |  |
| --- | --- |
| typedef struct node\_struct \* link;  struct node\_struct {  int item;  link next;  }; | typedef struct list\_struct \* list;  struct list\_struct {  link first;  int length;  }; |

A new node structure (intended to be used to create a list of lists) is defined as follows:

typedef struct coll\_node\_struct \* **coll\_link**;

struct coll\_node\_struct {

**list L;**

**coll\_link** next;

};

In your drawings, **show all the data as done in class** (including the list nodes, of type node\_struct). Use boxes for all member variables and write their value inside the box and their name outside the box.

Please find the drawing in the following pages

a) (7 points) Draw two nodes (of type coll\_node\_struct**)** that point to each other. For one of them L should be empty and for the other one L should be: 30->15->18 .

Please find the drawing in the following pages.

b) (3 points) Assume that an int is stored in 4 Bytes and a memory address is 8 Bytes. How much space will the above two nodes (and the data that they reference) occupy? That is, give the total space needed to store in memory what you drew above. **SHOW YOUR WORK**.

76 Bytes (Shown work in the following pages. Kindly look into it)