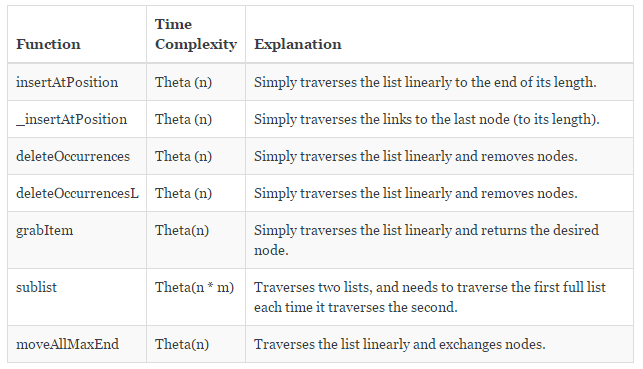
NAME: Jesse Daniel Mitchell ID: 1001449163

Task 1

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test case | Data/code | Does my code handle it? |
| 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 |
|  | A is NULL | list A = NULL;  result: fct returns NULL | Yes |
|  | A is empty | list A = newList();  result: fct returns NULL | Yes |
|  | pos\_list is empty | list pos\_list = NULL;  result: fct returns NULL | Yes |
|  | pos\_list is NULL | list pos\_list = newList(); result: fct returns NULL | Yes |
|  | 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 |
|  | Normal data  (as in hw writeup) | A: 15 -> 100 -> 7 -> 5 -> 100 -> 7 -> 30  pos\_list: 3 -> 0 -> 6 -> 4 | Yes |
|  | Repeated position | A: 5  pos\_list: 0 -> 0 -> 0  result: returns: 5-> 5-> 5 | Yes |
|  |  |  |  |
| **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 |
|  | V does not occur in A | A: 15 -> 100 -> 7 -> 5  V is 9,  Result: A does not change:  15-> 100-> 7-> 5 | Yes |
|  | Repeated consecutive occurrences | A: 15 -> 7 -> 7 -> 5  V is 7,  Result: A becomes:  15 -> 100 | Yes |
|  | A has one item and that is V | A: 7  V is 7  Result: A becomes Empty | Yes |
|  | A has only items with value V in it | A: 7->7-> 7  V is 7  Result: A becomes empty | Yes |
|  | A is NULL | A = NULL  Result: A is not changed | Yes |
|  | A is empty | A = newList()  Result: A is not changed | Yes |
|  |  |  |  |
| **insertAtPosition**  **(list A, Item val, int P)** | Normal data  (as in hw write-up) | A: 15 -> 100 -> 5 -> 100 -> 30 val = 12, P = 0  Result: A will become:  12-> 15-> 100-> 5-> 100-> 30 | Yes |
|  | A is NULL | A = NULL  Result: A is not changed | Yes |
|  | A is empty | A = newList()  Val = 12, pos = 0  Result: A will have item 12. | Yes |
|  | Position is greater than length of A | A: 2->3,val = 12, P = 5  Result: will insert at the end of A. A becomes:  2->3->12 | Yes |
|  | Position is negative | A: 2->3,val = 12, P = -2  Result: will insert at the beginning of A. A becomes:  12 -> 2 ->3 | Yes |
|  |  |  |  |
| **moveAllMaxAtEnd**  **(list A)** | A is NULL | A = NULL  Result: A is not changed | Yes |
|  | A is empty | A = newList()  Result: A is not changed | Yes |
|  | Normal data  (as in hw write-up) | A: 15 -> 100 -> 5 -> 100 -> 30  Result: A will become:  15 -> 5 -> 30 -> 100 -> 100 | Yes |
|  | A has one item | A: 7  Result: A does not change | Yes |
|  | 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 |
|  | MAX is on first position | A: 100-> 7->20  Result: A: 7->20->100 | Yes |
|  | MAX is on last position | A: 10-> 7->200  Result: A: 10->7->200 | Yes |

Task 2: (The answer for this task may also be written in the source code.)



Compile Instructions:

To compile and run, type:

gcc list\_hw.c instructor\_client.c -o instr  
instr   
  
   
or

sh compile.sh  
instr