|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NAME: \_JOSHUA ABUTO\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |  |  |  | ID:1001530342 | |
| Task 1: test cases for your code | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Function | Test case | Data/code | | | | | Does my code handle it? |
|  |  |  |  |  |  |  | Here: handle= does NOT crash |
| sublist(link A, link | Index out of bounds | A: 10 ->10 ->40 ->20 | | | | | No |
| pos\_list) |  | pos\_list: (**-7**) -> 3 or | | | | |  |
|  |  |  |  |  |  |  |  |
|  |  | pos\_list: | 3 -> **80000** -> 3 | | | |  |
|  |  |  | | |  |  |  |
|  |  | result: fct returns NULL | | | | |  |
|  | A is NULL | link A = NULL; | | | | | Yes |
|  |  | result: fct returns NULL | | | | |  |
|  | A is empty | link A = new\_list(); | | | | | Yes |
|  |  | result: fct returns NULL | | | | |  |
|  | pos\_list is empty | link pos\_list = NULL; | | | | | Yes |
|  |  | result: fct returns NULL | | | | |  |
|  | pos\_list is NULL | link pos\_list = newList(); | | | | | Yes |
|  |  | result: fct returns NULL | | | | |  |
|  | A is not modified by | A: 15 -> 100 -> 7 -> 5 -> 100 | | | | | Yes |
|  | sublist(…) | pos\_list: | 3 -> 0 ->2 | | |  |  |
|  | …. | result: A will still be : | | | | |  |
|  |  | 15 -> 100 -> 7 -> 5 -> 100 | | | | |  |
|  | Normal data | A: 15 -> 100 -> 7 -> 5 -> 100 - | | | | | Yes |
|  | (as in hw writeup) | > 7 -> 30 |  |  |  |  |  |
|  |  | pos\_list: 3 -> 0 -> 6 -> 4 | | | | |  |
|  | Repeated position | A: 5 |  |  |  |  | Yes |
|  |  | pos\_list: | 0 -> 0 -> 0 | | |  |  |
|  |  | result: returns: 5-> 5-> 5 | | | | |  |
|  |  |  | | | | |  |
| delete\_occurrences | Normal data, V is in A | A: 15 -> 100 -> 7 -> 5 -> 100 - | | | | | No |
| (link A, int V) | (as in hw write-up) | > 7 -> 30 |  |  |  |  |  |
|  |  | V is 7, |  |  |  |  |  |
|  |  | Result: A will become: | | | | |  |
|  |  | 15-> 100-> 5 -> 100 -> 30 | | | | |  |
|  | V does not occur in A | A: 15 -> 100 -> 7 -> 5 | | | | | Yes |
|  |  | V is 9, |  |  |  |  |  |
|  |  | Result: A does not change: | | | | |  |
|  |  | 15-> 100-> 7-> 5 | | | |  |  |
|  | Repeated consecutive | A: 15 -> 7 -> 7 -> 5 | | | | | Yes |
|  | occurrences | V is 7, |  |  |  |  |  |
|  |  | Result: A becomes: | | | | |  |
|  |  | 15 -> 5 |  |  |  |  |  |
|  | A has one item and | A: 7 |  |  |  |  | Yes |
|  | that is V | V is 7 |  |  |  |  |  |
|  |  | Result: A becomes Empty | | | | |  |
|  | A has only items with | A: 7->7-> 7 | | | | | Yes |
|  | value V in it | V is 7 |  |  |  |  |  |
|  |  | Result: A becomes empty | | | | |  |
|  | A is NULL | A = NULL |  |  |  |  | Yes |
|  |  | Result: A is not changed | | | | |  |
|  | A is empty | A = new\_list() | | | | | Yes |
|  |  | Result: A is not changed | | | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| swap\_first\_third(link A) | A is NULL | A = NULL |  |
|  |  | Result: A is not changed |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | Yes |
|  | A is Empty | Result: A not changed | Yes |
|  | A has the same number | A(0) = 7  A(2) = 7  Result: no change notice but swap occurs | Yes |
|  | A has a length of two | Result: swap first and last position | No |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

CODE & DRAWING for swap\_first\_third (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 :

|  |  |  |
| --- | --- | --- |
| Let N = |A| (N is the size of list A) |  |  |
| swap\_first\_third(link A): | T(N) = N + 4 + 1 | |
| insertion\_sort (link A) | T(\_\_\_\_\_\_) = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Updated 2/12/19 |
| delete\_occurrences(link A, int V) | T(N,V) = 3+N\*(1+2)+1 | |
| sublist(link A, link pos\_list) | T(N,P) = 5+P\*(1+P\*(5)+5)+4 | |

Task 3 (10 points) Given:

A new node structure (intended to be used to create a list of lists) is defined in the table below (using struct node):

struct node { int item;

struct node \* next;

};

struct coll\_node {

**link Ld;** // NOTE: Ld must be represented with a dummy starting node. **struct coll\_node \*** next;

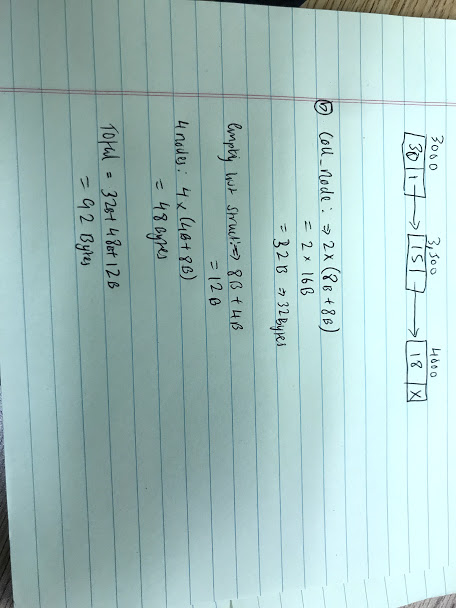
};

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

1. (7 points) Draw two nodes (of type struct coll\_node**)** that point to each other. For one of them Ld should be empty (but not NULL) and for the other one Ld should be: 30->15->18 . Use the representation with a DUMMY node for any normal list, Ld, part of nodes of type struct coll\_node.

A close up of text on a whiteboard

Description automatically generated

1. (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**.