HOMEWORK 3

NAME: ELSY FERNANDES

STU ID: 1001602253

SECTION 002

```
void deleteOccurrences(list A, Item V)
{
   if (A == NULL) {
      printf("List A is NULL\n"); //Check if List A is NULL
      return;
   }
   int ListLength = getLength(A);
   if (ListLength == 0) { // Check if List A is empty
      printf("List A is empty \n");
   }
   else {
      int i;
      link current = getFirst(A); //first link                    //Get the first link
      link previous = NULL;
      int ListLength = getLength(A); //Get the Length of the list
      for (i = 0; i < ListLength; i++) {
         Item TestElement = getLinkItem(current); // Get the First item of the link
         link nextLinkNew = getLinkNext(current); // Get the Next Link
         if (TestElement == V) { //Check if Test Element is equal to item to be deleted
            link temp;
            if (i == 0) { //Removes first link
               temp = removeFirst(A);
               destroyLink(temp);
            }
            else {
               temp = removeNext(A, previous); //Removes link coming after previous
               destroyLink(temp);
            }
            current = nextLinkNew; //Repeat for i =0 to List length
         }
         else {
            previous = current;
            current = nextLinkNew;
         }
      }

   }
   return;
}
```

Constant C_1

It can delete all the nodes in the list Hence NC_2

$$T(N) = C_1 + NC_2$$

$$T(N) = \Theta(N)$$

```
list sublist(list A, list pos_list)
{
   if (A == NULL) {
      printf("list A is NULL\n");                    //Check if the list A is NULL
      return NULL;
   }
   int listLength = getLength(A);
   if (listLength == 0) {
      printf("List A is empty\n");                   //Check if the list A has zero length
      return NULL;
   }
   if (pos_list == NULL) {
      printf("List A is NULL \n");                   //Check if the pos_list is NULL
      return NULL;
   }

   if (pos_list->length == 0) {                      //Check if the list pos_list has zero length
      printf("List A is empty \n");
      return NULL;
   }
   else {
      int i;
      list subList = newList();                      //Create new sublist to store the values
returned from the function
      link sublistLink = NULL;
      link newLinkToAdd;
      link startLink = getFirst(A);                  //Get first link of list A
      link currentLink, nextLink;
      link previousLinkAddedToSublist;
      link currentLinkForPosList = getFirst(pos_list);        //Get first link of pos_list
      int posListLength = getLength(pos_list);               //Get length of pos_list
      int count = 0;
      Item itemAtRequiredIndex;
      Item itemToGet = getLinkItem(currentLinkForPosList);    //Get first item of Pos_list

      currentLink = startLink;
      for (i = 0; i < posListLength; i++) {
         if (itemToGet > listLength) {                //Handled for Index out of bound test
case
            destroyList(subList);
            return NULL;
         }
         while (count != itemToGet) {
```

Constant Time C_1

constant time C_2

goes over all the values of pos_list n times

Subtot - $\frac{N(N+1)}{2}$

```
            nextLink = getLinkNext(currentLink);            //get the value from pos_list
            count++;
            currentLink = nextLink;
          }
          newLinkToAdd = newLink(getLinkItem(currentLink), NULL);   //if lengthof sublist
==0 Insert the value returned from the list A based on the pos_list value to the new
sublist
          if (getLength(subList) == 0) {
            insertAtBeginning(subList, newLinkToAdd);
          }
          else {
            insertLink(subList, previousLinkAddedToSublist, newLinkToAdd); //else insert
to next node
          }
          previousLinkAddedToSublist = newLinkToAdd;
          currentLinkForPosList = getLinkNext(currentLinkForPosList);
          if (currentLinkForPosList != NULL) {
            itemToGet = getLinkItem(currentLinkForPosList);
          }
          count = 0;
          currentLink = startLink;
        }
        return subList;
    }

}
```

If pos_list 5→0→6→4

If A→15→100→7→5→100
→7→30

For pos_list 5
it checks for all the
values of A
0,1,2,3,4,5

Total = $T(N)$

$$C_1 + C_2 + \frac{n(n+1)}{2} C_3$$

$$\frac{n(n+1)}{2}$$

Hence $T(N) = \Theta(N^2)$

```
void swapFirstThird(list A)
{
   int listLength = getLength(A);                    //Get List length
   link previous = NULL;

   if (A == NULL) {
      printf("List A is Null \n");                   //Check if A is NULL
   }
   else if (listLength == 0) {                       //Check if list length is zero
      printf("List A is empty\n");
   }

   else if (listLength > 2) {
      link current = getFirst(A);
      link secondLink = getLinkNext(current);
      link thirdLink = getLinkNext(secondLink);
      link fourthLink = getLinkNext(thirdLink);      //If list length is greater than 2 swap
first and third
      setLinkNext(current, fourthLink);
      setLinkNext(secondLink, current);
      setLinkNext(thirdLink, secondLink);
      setFirst(A, thirdLink);
   }
   else if (listLength == 2) {
      link current = getFirst(A);
      link temp = NULL;
      link secondLink = getLinkNext(current);        //If list length is equal to 2 swap
first and second
      temp = current;
      A->first = secondLink;
      link newFirst = getFirst(A);
      setLinkNext(newFirst, temp);
   }
}
```

C_1

C_2

1 time

$T(N) = C_1 + C_2 + 1$

Hence $T(N) = \Theta(1)$

```
void moveAllMaxAtEnd(list A)
{
   int listLength = getLength(A);
   if (A == NULL) {
      printf("List A is NULL\n");
      return;
   }
   else if (listLength == 0) {
      printf("List A is empty \n");
   }
   else if (listLength > 1) {
      int i;
      int j;
      int listLength = getLength(A);
      link current = getFirst(A);
      Item testItem;
      Item newTestItem;
      int maxValue = 0;
      link previous = NULL;
      int countOfOccurences = 0;
      for (i = 0; i < listLength; i++) {
         Item testItem = getLinkItem(current);
         if (testItem > maxValue) {
            maxValue = testItem;
         }
         link nextLink = getLinkNext(current);
         current = nextLink;
      }
      current = getFirst(A);
      for (i = 0; i < listLength; i++) {
         Item testItem = getLinkItem(current);
         if (testItem == maxValue) {
            countOfOccurences = countOfOccurences + 1;
         }
         link nextLink = getLinkNext(current);
         current = nextLink;
      }

      if (countOfOccurences < listLength) {
         link linksRemoved[countOfOccurences];
         link nextCurrent = getFirst(A);
         int arrayPointer = 0;
         for (i = 0; i < listLength; i++) {
```

C_1

$\rightarrow 1$

$\rightarrow$ for Every Element of testItem it checks Max Value $\rightarrow N$

For every element of testItem Compared with max Value $\rightarrow N$

$\rightarrow$ Count of occurance $\rightarrow N$

```
            Item testItem = getLinkItem(nextCurrent);
            link nextCurrentLink = getLinkNext(nextCurrent);
            if (testItem == maxValue) {
               if (previous == NULL) {
                  A->first = nextCurrentLink;
               }
               else if (nextCurrentLink != NULL) {
                  setLinkNext(previous, nextCurrentLink);
               }
               linksRemoved[arrayPointer] = nextCurrent;
               arrayPointer++;
            }
            previous = nextCurrent;
            nextCurrent = nextCurrentLink;
         }
         for (i = 0; i < countOfOccurences; i++) {
            link maxLink = linksRemoved[i];
            setLinkNext(maxLink, NULL);
            setLinkNext(previous, maxLink);
            previous = maxLink;
         }
      }
   }
}
```

→

→ N

$$T(N) = C_1 + N + N + N + N$$

$$T(N) = \Theta(N)$$

NAME: ELSY FERNANDES ID: 1001602253

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 -> 5	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
swapFirstThird (list A)	STUDENTS must give the special cases for this function. (Add or remove rows as needed.)	STUDENTS must give the example data	
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~~ Yes

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 :

TEST CASES FOR FIRST AND THIRD SWAP

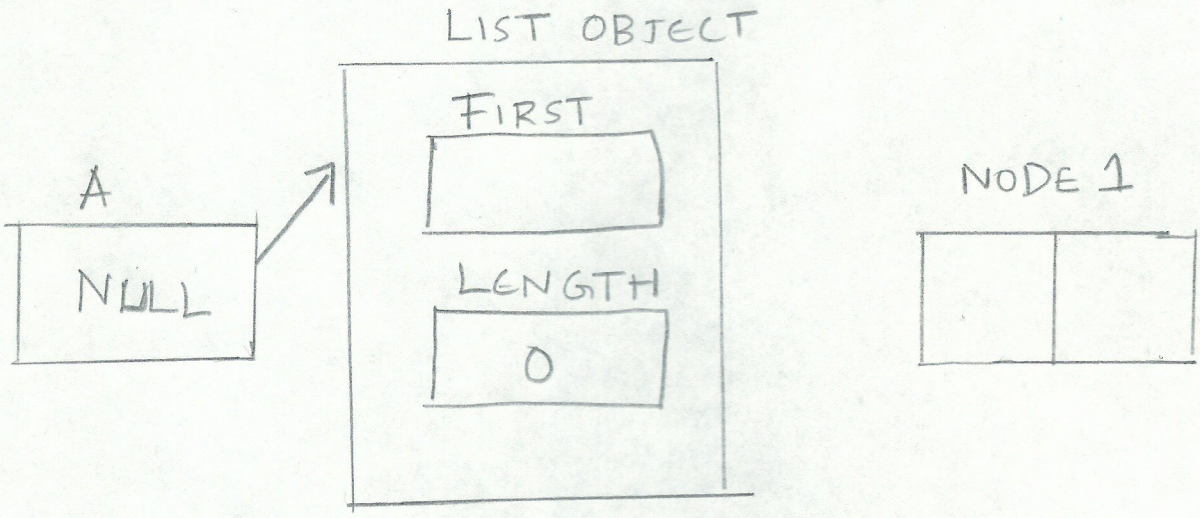
Function	Testcase	Data/Code	Does my code Handle?
SwapFirstThird(A);	Lis A is NULL	A=NULL ~~Result Not Changed~~ Function Return NULL	Yes
	List A is Empty	A = newList(); Function Returns NULL	Yes
	List A has One Element	A: 15 Result: List is not Changed	Yes
	List A has 2 Elements	A: 15→30 Result: 30→15	Yes
	List A has 3 Elements	A→15→30→20 Result 20→30→15	Yes
	Lis A has more than 3 Elements	A→15→30 →20→40→ 50→16 Result: 20→30→15 →40→50→16	Yes

CODE and DRAWING FOR swapFirstThird(list A)

```
void swapFirstThird(list A)
{
   int listLength = getLength(A); //Get List Length
   link previous = NULL;

   if (A == NULL) { //Check if List A is NULL
      printf("List A is NULL\n");
      return;
   }
   else if (listLength == 0) { //Check if List A is empty
      printf("List A is empty \n");
   }

   else if (listLength > 2) {
      link current = getFirst(A);
      link secondLink = getLinkNext(current);
      link thirdLink = getLinkNext(secondLink); //If Length > 2 ,Swap first and third
      link fourthLink = getLinkNext(thirdLink);
      setLinkNext(current, fourthLink); -> (5)
      setLinkNext(secondLink, current); -> (6)
      setLinkNext(thirdLink, secondLink); -> (7)
      setFirst(A, thirdLink); -> (8)

   }
   else if (listLength == 2) {
      link current = getFirst(A);
      link temp = NULL; — (1)
      link secondLink = getLinkNext(current); //If Length == 2 ,Swap first and second
      temp = current; — (2)
      A->first = secondLink; — (3)
      link newFirst = getFirst(A);
      setLinkNext(newFirst, temp); — (4)
      destroyLink(temp);
   }
}
```

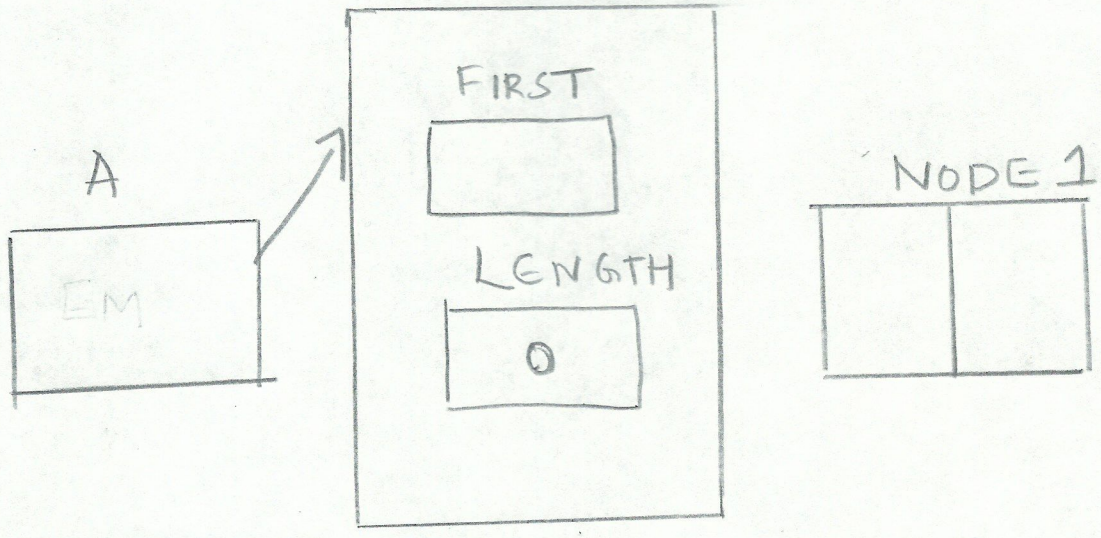
Drawing for SwapFirst-third(listA)

Test Case 1: A is NULL



→ Returns NULL

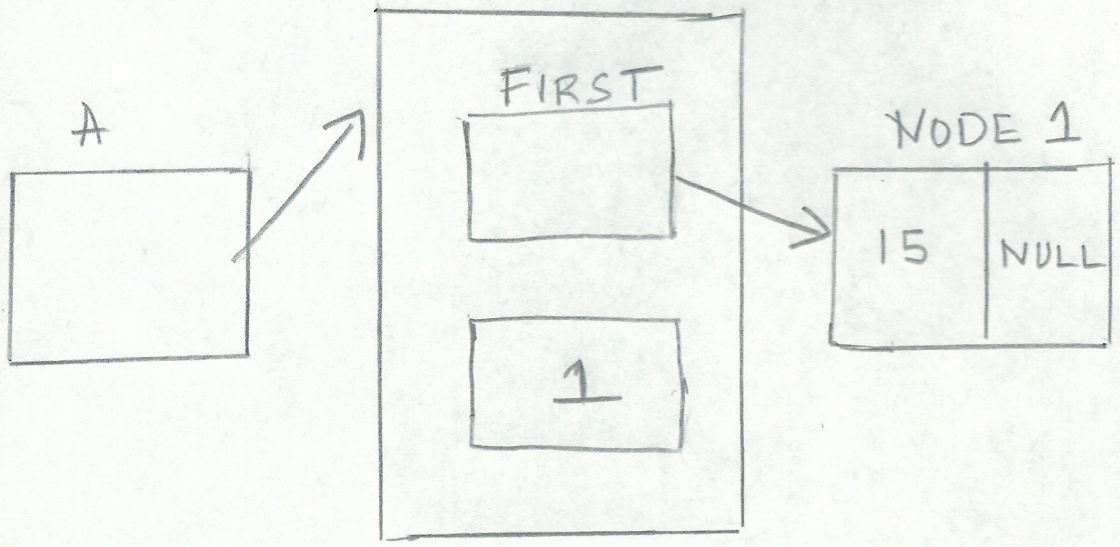
Test Case 2: A is Empty



→ Returns NULL

TESTCASE3: A has One item

List A → 15

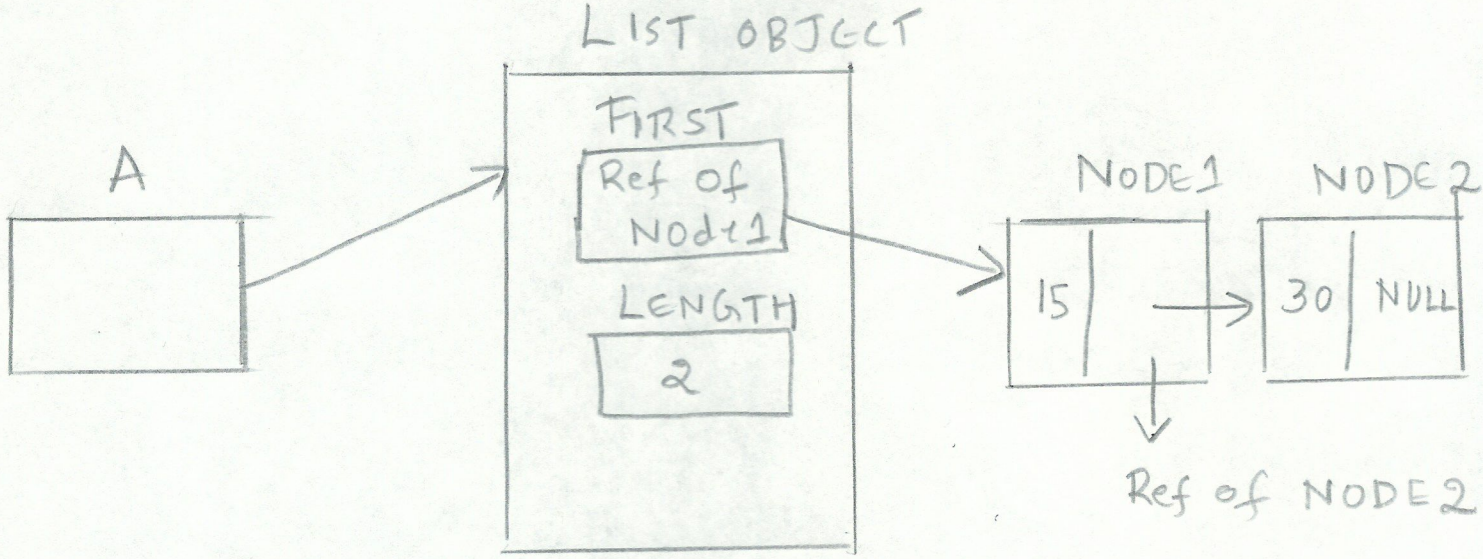


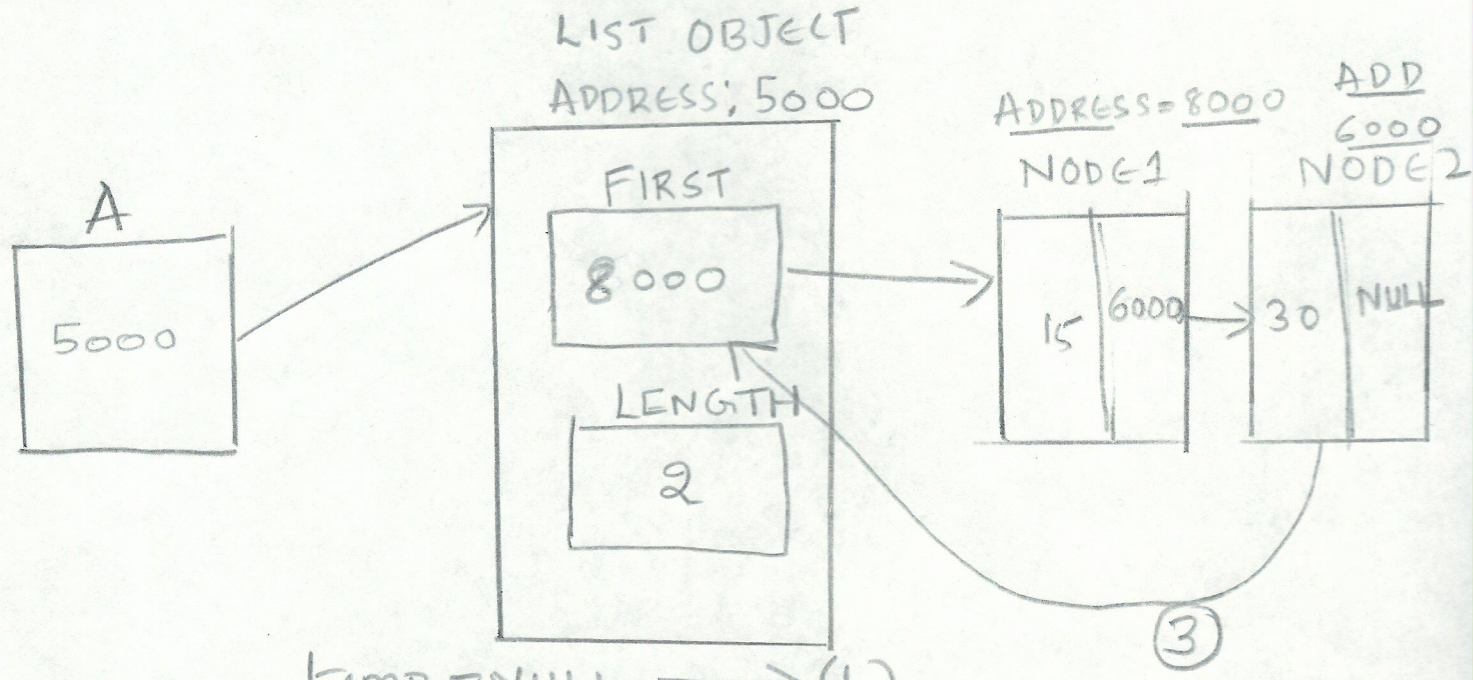
Result: Since it has only one Element No swapping is done

Returns 15

TEST CASE : A has 2 items

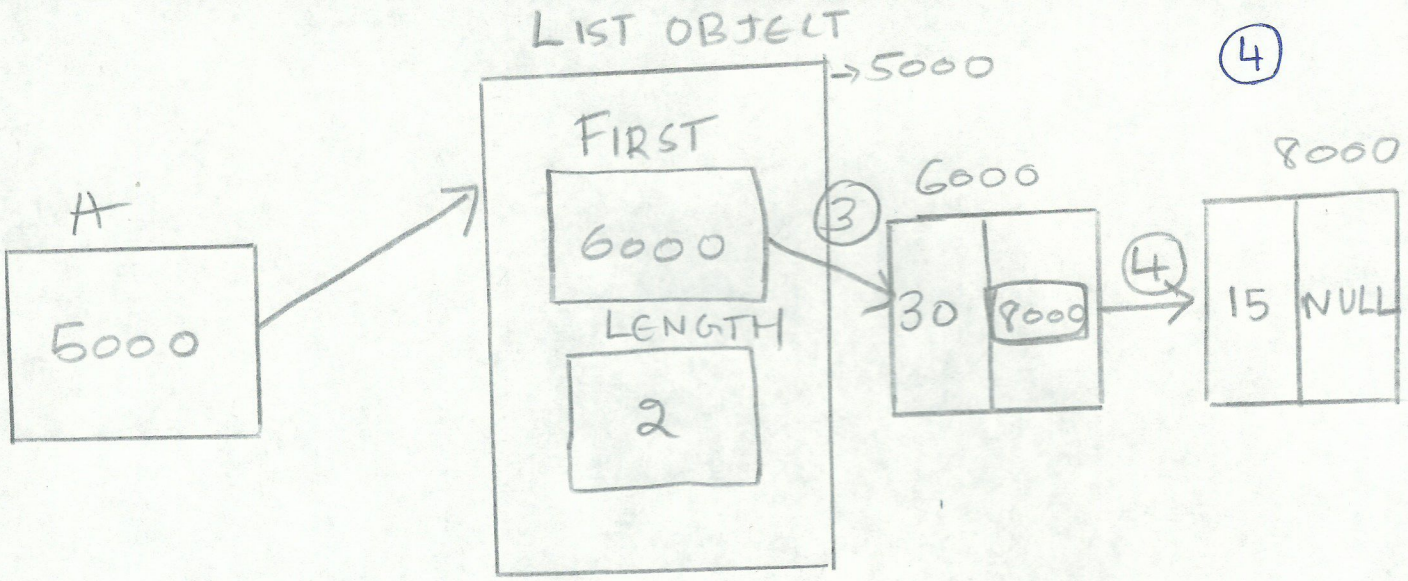
List A = 15 → 30





temp = NULL ⟶ (1)

temp = A → first = 8000 ⟶ Code (2)



temp = 8000

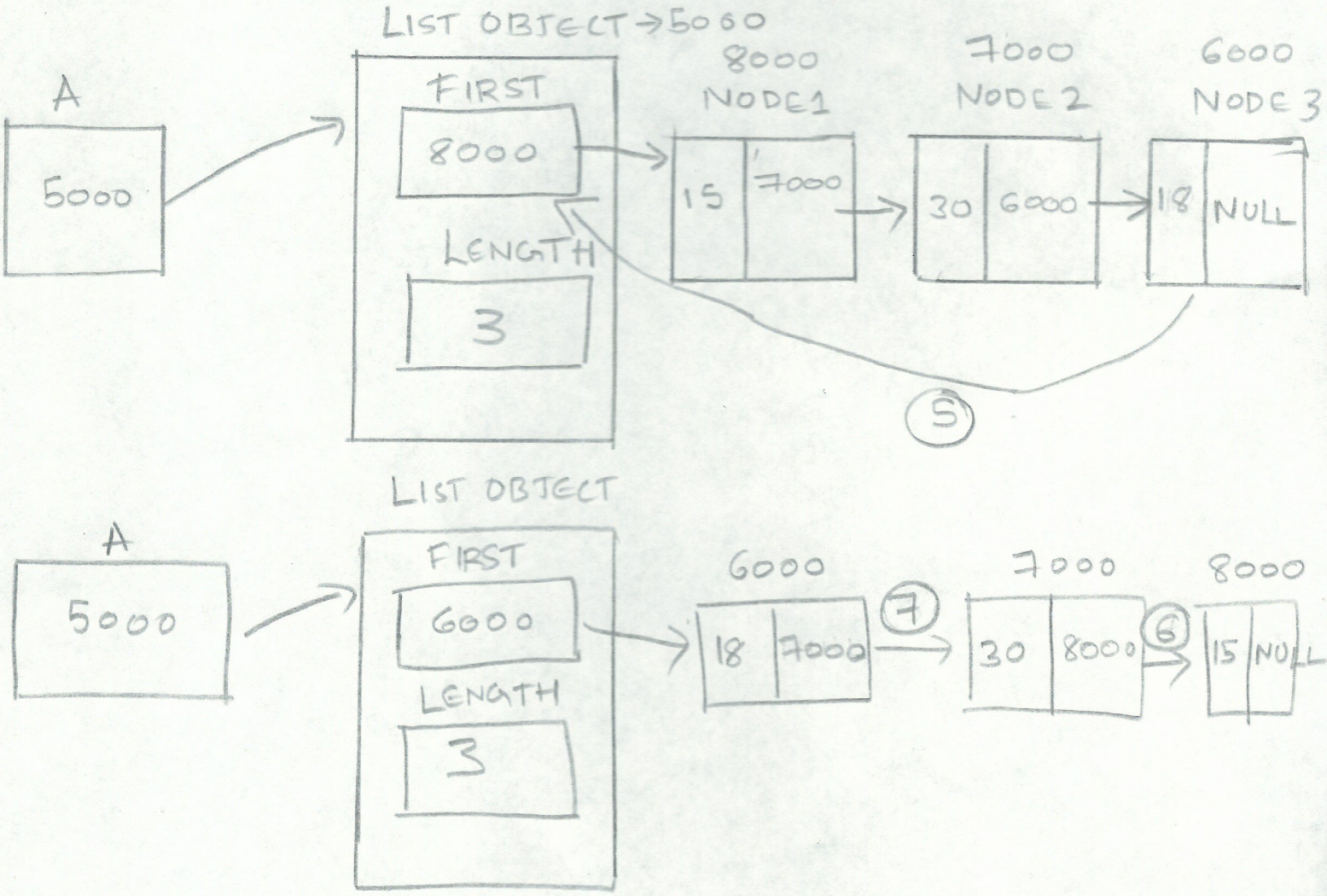
setLinkNext(newfirst, temp) → (4)

TESTCASE 5 & 6

(5)

→ A has 3 items

→ A has more than 3 items



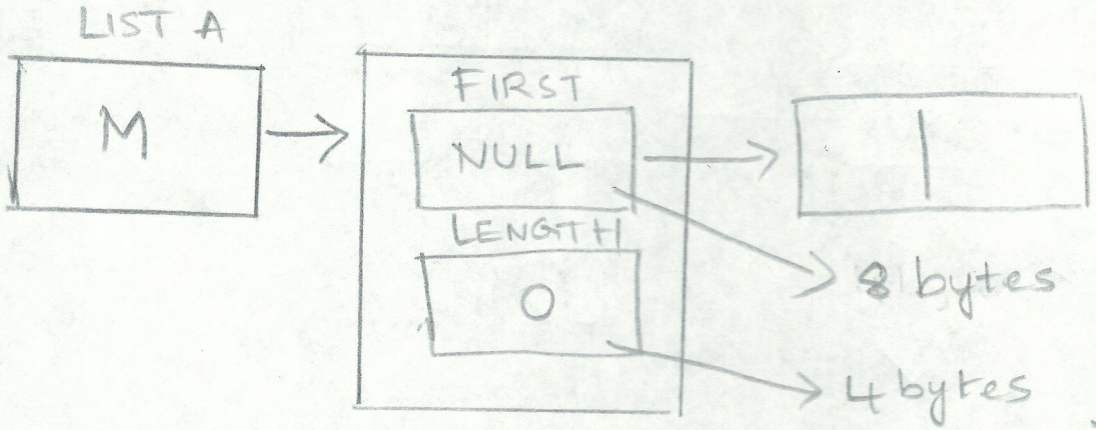
TASK 3

A new node structure is defined as follows.

```
Typedef Struct coll_node_Struct *Coll_link;
        Struct coll_node_Struct {
  List L;
  Coll_link next;
  };
```

LIST 1

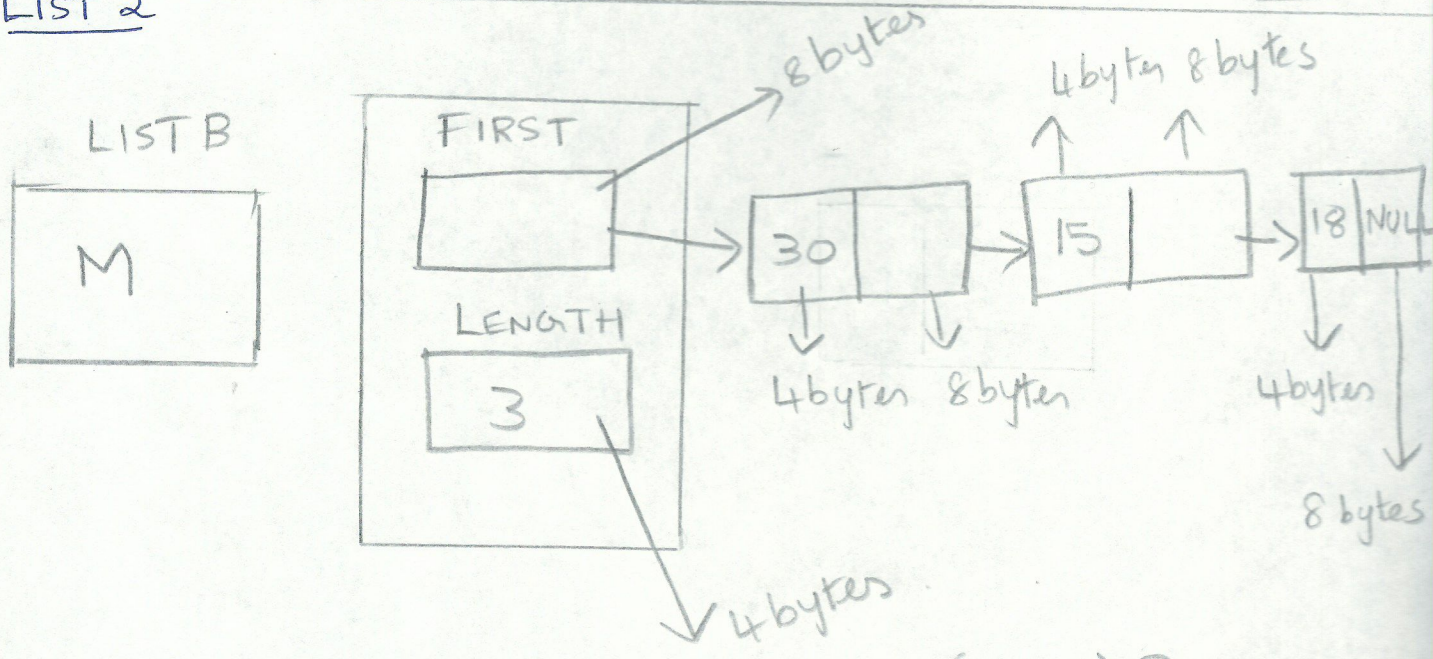
M → Memory



= 8 + 4

= 12 — ①

LIST 2



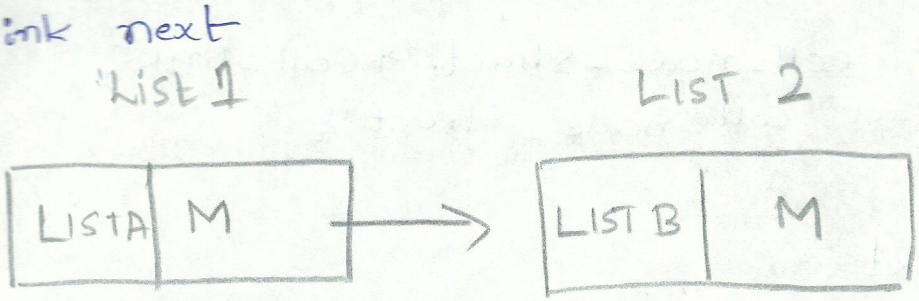
8 + 4 + (4 + 8) 3

12 + 12 × 3 = 12 + 36

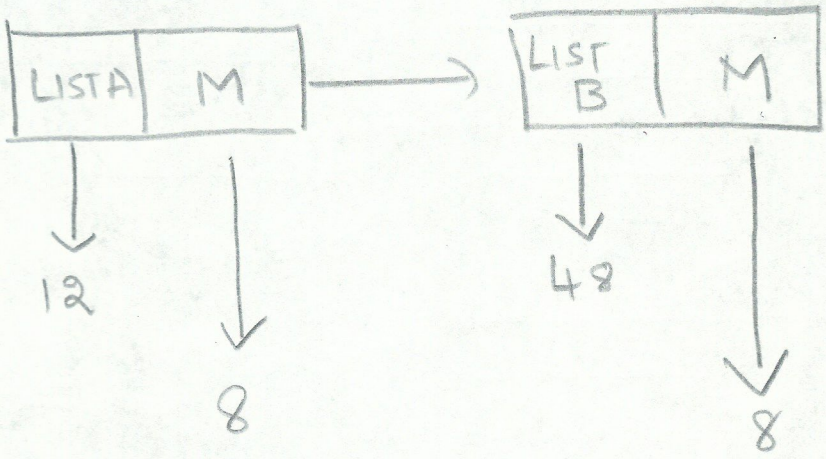
= 48 — ②

M → Memory

Coll_link next



B) Total Space could be → substitute (1) & (2)



$= 12 + 8 + 48 + 8$

$= 20 + 56$

$= \underline{\underline{76}}$ bytes

INSTALLATION INSTRUCTIONS

1) Copy all the files to the omega server

   1) list_hw.c
   2) list_hw.h
   3) instructor client.c

2) Compile

```
gcc -o instr instructor_client.c list_hw.c
```

3) Execute

```
./instr
```

4) To check memory leak

```
Valgrind --leak-check=yes ./instr
```