**Question 1 .**

**Write a function “insert\_any()” for inserting a node at any given position of the linked list. Assume position starts at 0.**

**Solution ;**

// function to insert a Node at required position

void insert\_any (Node\*\* current, int pos, int data)

{

// This condition to check whether the

// position given is valid or not.

if (pos < 1 || pos > size + 1)

cout << "Invalid position!" << endl;

else {

// Keep looping until the pos is zero

while (pos--) {

if (pos == 0) {

// adding Node at required position

Node\* temp = getNode(data);

// Making the new Node to point to

// the old Node at the same position

temp->next = \*current;

// Changing the pointer of the Node previous

// to the old Node to point to the new Node

\*current = temp;

}

else

// Assign double pointer variable to point to the

// pointer pointing to the address of next Node

current = &(\*current)->next;

}

size++;

}}

**Question 2 .**

**Write a function “delete\_beg()” for deleting a node from the beginning of the linked list.**

**Solution ;**

void delete\_beg() ()

{

struct node \*toDelete;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

head = head->next;

printf("\nData deleted = %d\n", toDelete->data);

/\* Clears the memory occupied by first node\*/

free(toDelete);

printf("SUCCESSFULLY DELETED FIRST NODE FROM LIST\n");

}

}

**Question 3.**

**Write a function “delete\_end()” for deleting a node from the end of the linked list.**

**Solution;**

void delete\_end()

{

struct node \*toDelete, \*secondLastNode;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

secondLastNode = head;

/\* Traverse to the last node of the list \*/

while(toDelete->next != NULL)

{

secondLastNode = toDelete;

toDelete = toDelete->next;

}

if(toDelete == head)

{

head = NULL;

}

else

{

/\* Disconnect link of second last node with last node \*/

secondLastNode->next = NULL;

} free(toDelete);

printf("SUCCESSFULLY DELETED LAST NODE OF”);

**Question 4 .**

**In the Binary Search algorithm, it is suggested to calculate the mid as beg + (end - beg) / 2 instead of (beg + end) / 2. Why is it so?**

**Solution;**

beg + (end - beg) / 2

reads as:

mid equals start plus half of the length.

whereas:

(beg + end) / 2

reads as:

mid equals half of start plus end

Which does not seem as clear as the first, at least when expressed like that.

**Question 5.**

**Write the algorithm/function for Ternary Search.**

**Solution ;**

int ternary\_search(int l,int r, int x)

{

if(r>=l)

{

int mid1 = l + (r-l)/3;

int mid2 = r - (r-l)/3;

if(ar[mid1] == x)

return mid1;

if(ar[mid2] == x)

return mid2;

if(x<ar[mid1])

return ternary\_search(l,mid1-1,x);

else if(x>ar[mid2])

return ternary\_search(mid2+1,r,x);

else

return ternary\_search(mid1+1,mid2-1,x);

}

return -1;

}