Assignment 4

**Question 1**

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

Ans :

void insert\_pos(int pos)

{

int I;

NODE \* curptr,\*newnode=NULL;

newnode=(NODE\*)malloc(sizeof(NODE));

printf("enter USN\n");

scanf("%s",newnode->usn);

printf("enter phone number\n");

scanf("%s",newnode->phno);

if(start==NULL)

{

newnode->link=start;

start=newnode;

}

else

{

curptr=start;

for(i=2;i<=pos-1;i++)

{

curptr=curptr->link;

if(curptr == NULL)

break;

}

if(curptr!=NULL)

{

newnode->link=curptr->link;

curptr->link=newnode;

}

else

printf(“Unable to insert data”);

}

}

**Question 2**

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

Ans:

void del\_beg()

{

NODE\* curptr=NULL;

if(start==NULL)

printf("\n LIST EMPTY\n");

else

{

curptr=start;

start=start->link;

printf("the information is \n");

printf("\n NAME \tPHONENUMBER\n");

printf(" %s\t%s\n" ,curptr->name,curptr->phno);

free(curptr);

}

}

**Question 3**

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

Ans:

void delete\_end()

{

NODE \*curptr=NULL,\*nextcur=NULL;

if(start==NULL)

printf("\n LIST EMPTY\n");

else if(start->link==NULL)

free(start);

else

{

curptr=start;

nextcur=curptr;

while(curptr->link!=NULL)

{

nextcur=curptr;

curptr=curptr->link;

}

nextcur->link=NULL;

free(curptr);

}

}

**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?

Ans: calculating mid this way is ineffective. Let’s take an example.  
Let us take integers from an integer low to an integer high (both included).

* for low = 3 and high = 11, the number of elements (#elements) = 9 So there is only 1 mid, i.e., 7  
  \* Both formulae have computed the mid correctly
* for low = 3 and high = 10, #elements = 8 So there are 2 mids, 6 (lower mid) and 7 (higher mid)  
  \* Both formulae have computed the lower mid correctly
* for low = -11 and high = -3, #elements = 9 So there is only 1 mid, i.e., -7  
  \* Both formulae have computed the mid correctly
* for low = -10 and high = -3, #elements = 8 So there are 2 mids, -7 (lower mid) and -8 (higher mid)  
  \* The formula(low + high) / 2 has failed to compute the lower mid correctly but the other formula has computed it correctly.

So, we should always use the below formula to compute lower mid as it is much more reliable:

int mid = low + ((high - low)/2);

**Question 5**

Write the algorithm/function for Ternary Search.

Ans:

int ternarySearch(int l, int r, int key, int ar[])

{

    if (r >= l) {

        // Find the mid1 and mid2

        int mid1 = l + (r - l) / 3;

        int mid2 = r - (r - l) / 3;

        // Check if key is present at any mid

        if (ar[mid1] == key) {

            return mid1;

        }

        if (ar[mid2] == key) {

            return mid2;

        }

        // Since key is not present at mid,

        // check in which region it is present

        // then repeat the Search operation

        // in that region

        if (key < ar[mid1]) {

            // The key lies in between l and mid1

            return ternarySearch(l, mid1 - 1, key, ar);

        }

        else if (key > ar[mid2]) {

            // The key lies in between mid2 and r

            return ternarySearch(mid2 + 1, r, key, ar);

        }

        else {

            // The key lies in between mid1 and mid2

            return ternarySearch(mid1 + 1, mid2 - 1, key, ar);

        }

    }

    // Key not found

    return -1;

}