```c
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 100


typedef struct node {
    int data;
    struct Node* next;

}Node;

Node* create(int val)
{
   Node* n = (Node*)malloc(sizeof(Node));

   n->data = val;
   n->next = NULL;

   printf("\n\nFunction Create Head Node: %u Address of Head Node %d Data of Head Node,
%u Next of Head Node", n, n->data, n->next);
   return n;
}

void PrintList(Head)
{
   int i=0;
   // If more nodes are added
   Node *temp1 = (Node*)malloc(sizeof(Node));
   if (Head==NULL)
        printf("\n\nLinked List is empty...");
   else
   {

       temp1=Head;
       printf("\n\nFunction PrintList:");

       while(temp1 != NULL)
       {
           printf("\nNode %d: %u Addr, %d Data, %u Next Addr", i++,temp1, temp1->data,
temp1->next);
           temp1 = temp1->next;
       }
    }
    free(temp1);
}

Node *Insertatend(Node *Head, int val)
{

   //create a node
   Node *n = (Node*)malloc(sizeof(Node));
   n->data = val;
   n->next = NULL;

// Only single node after head
```

```
//   head->next = n;
//   printf("\n\n Function InsertAtEnd: %u Address of Head->next Node, %d Data of Head-
>next Node, %u Next of Head->next Node", n, n->data, n->next);

   // If more nodes are added then require to reach the last node
   if (Head!=NULL)
   {

   Node *temp = (Node*)malloc(sizeof(Node));
   temp=Head;
   printf("\n\nFunction InsertAtEnd:Loop From Head till End:\n%u Address of Head Node, %d
Data of Head Node,  %u Next of Head Node", temp, temp->data, temp->next);

   while(temp->next != NULL)
   {
       printf("\nLOOP: %u Address of Node, %d Data of Node,  %u Next of Node", temp, temp-
>data, temp->next);
       temp = temp->next;
   }

    temp->next = n;
    }
else
    Head=n;
    printf("\n\nInserted Node with %d data and its next %u", n->data, n->next);
//  free(n);
    return Head;
}
Node *InsertAtFront(Node *Head, int val)
{
   //create a new node n
   Node *n = (Node*)malloc(sizeof(Node));
   n->data = val;
   n->next=Head;
   printf("\n\nInsertAtFront: Inserted node as Head Node WITH %u new address, %u data, %u
next of Head Node", n, n->data, n->next);

   Head=n;
//   free(n);
   return Head;
}
Node *DeleteFromFront(Node *Head)
{

   //create a new node n
   Node *n = (Node*)malloc(sizeof(Node));
   printf("\n\nFunction DeleteFromFront: Delete Head Node: %u Address,  %d Data, %u Next",
 Head, Head->data, Head->next);
   n=Head;
   if(n!=NULL)
   {
        if (n->next !=NULL)
        {
            n=n->next;
        }
```

```
        else
            n=NULL;

    }
    Head=n;
    free(n);
    return Head;
}

Node *DeleteFromEnd(Node *Head)
{

    printf("\n\nFunction DeleteFromEnd:");
    Node *temp = (Node*)malloc(sizeof(Node));
    temp=Head;

    Node *prev = (Node*)malloc(sizeof(Node));
    if (temp->next==NULL)
     {
         temp=NULL;
         Head=temp;
     }
     else
     {

         while((temp->next) != NULL)
         {
            prev=temp;
            temp = temp->next;
         }
         temp=prev;
         temp->next=NULL;

     }

   // free(temp);
    return Head;
}

void main()
{
   Node* Head = create(10);
   printf("\n\tsize of Head node: %d Data and %d next = %d",sizeof(Head->data),
sizeof(Head->next),sizeof(Node));

   Head=DeleteFromFront(Head);
   PrintList(Head);

   Head=Insertatend (Head, 20);
   PrintList(Head);

   Head=DeleteFromEnd(Head);
   PrintList(Head);

   Head=Insertatend (Head, 25);
```

```
 PrintList(Head);

 Head=InsertAtFront(Head,15);
 PrintList(Head);

 Head=DeleteFromEnd(Head);
 PrintList(Head);

 Head=InsertAtFront(Head,5);
 PrintList(Head);

 Head=DeleteFromEnd(Head);
 PrintList(Head);

}
```