```
#include <stdio.h>
#include <stdlib.h>

typedef struct node *Nodeptr;
struct node{
   int data;
   Nodeptr next;
};

Nodeptr getnode(){
   Nodeptr temp;
   temp = (Nodeptr) malloc(sizeof(struct node));
   return temp;
}

int IsEmpty(Nodeptr first){
   return (first==NULL ? 1:0);
}

Nodeptr InsertFront(Nodeptr first, int x){
   Nodeptr temp;

   temp= getnode();
   temp->data = x;
   temp->next = NULL;

   temp->next = first;
   //first = temp;

   return temp;
}
```

```
void Display(Nodeptr first){
   Nodeptr temp;

   if (IsEmpty(first)){
      printf("Empty List");
      return;
   }
   temp=first;
   printf("Contents of List : \n");
   while(temp){
      printf("%d\n",temp->data);
      temp = temp->next;
   }
}
```

```
Nodeptr InsertLast(Nodeptr first, int x){
   Nodeptr temp, rear;

   temp= getnode();
   temp->data = x;
   temp->next = NULL;

   if (IsEmpty(first)){
      first =temp;
   }
   else{
      rear = first;
      while(rear->next)
         rear = rear->next;
      rear->next = temp;
   }
   return first;
}

int DeleteFront(Nodeptr *first){
   Nodeptr temp;
   int x;

   if (IsEmpty(*first)){
      printf("Empty List\n");
      return -999;
   }
   temp = *first;
   x = temp->data;
   *first = (*first)->next;
   free(temp);
   return x;
}
```

```
Nodeptr DeleteLast(Nodeptr first){
   Nodeptr temp, prev;
   if (IsEmpty(first)){
      printf("Empty List\n");
      return first;
   }
   temp = first;
   prev = NULL;
   while(temp->next){
      prev = temp;
      temp = temp->next;
   }
   if (prev == NULL )
      first = NULL;
   else
      prev->next = NULL;
   free(temp);
   return first;
}
```

```
Nodeptr InsertAfter(Nodeptr first, int x, int y){

Nodeptr temp, prev;
//search for y
prev= first;
while(prev){
  if (prev->data == y)
     break;
   prev=prev->next;
}
if (prev==NULL){ //y not found
   printf("%d Not .. returning", y);
   return first;
}

//y is found, insert x

temp = getnode();
temp->data = x;
temp->next = NULL;

temp->next = prev->next;
prev->next = temp;

return first;
}


Nodeptr Reverse(Nodeptr first){
  Nodeptr p,q,r;
  p= first;
  q=NULL;
  while(p){
     r=q;
     q=p;
     p=p->next;
     q->next = r;
  }
  return q;
}
```

```
Nodeptr Delete(Nodeptr first, int x){
   Nodeptr trail = NULL;
   Nodeptr temp = first;

   while (temp->data != x)  {
      trail = temp;
      temp = temp->next;
   }
   if (temp == NULL){
      printf("%d Not found",x);
      return first;
   }
   if (trail == NULL){
      first = first->next;
   }
   else{
      trail->next = temp->next;
   }
   free(temp);
   return first;
}
```

```
int main()
{

   Nodeptr first = NULL;

   first=InsertFront(first,20);

   first=InsertFront(first,10);

   Display(first);

   first = InsertLast(first,30);

   first = InsertLast(first,50);

   //first=Delete(first,20);
   //Display(first);
  // first=Delete(first,10);
   Display(first);
   int x = DeleteFront(&first);
   printf("%d Deleted\n", x);
   x = DeleteFront(&first);
   printf("%d Deleted\n", x);
   //first = DeleteLast(first);
   //first = DeleteLast(first);

   //first = Reverse(first);

   //first = InsertAfter(first,40, 60 );

   Display(first);
   return 0;
}
```