```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

typedef union {
  int   iValue;
  char  cValue;
  char* cp;
  char  cbuff[20];
} NodeData;     // linked_node data element

typedef struct {
  void *prev;
  void *next;
} LinkInfo;     // has info on links to and from the node

typedef struct  {
  NodeData data;
  LinkInfo linkTo;
} Node;   // linked list node element

Node *lhead = NULL;
Node *ltail = NULL;

void create();
void insert();
void ndelete();
void search();
void ndisplay();
void find();

int main()
{
  int choice, n;

  printf("\n Enter the initial no. of inputs:");
  scanf("%d",&n);

  int i;
  for (i=1;i<=n;i++)
    create();
  printf("\n1.Insert\n2.Delete\n3.Display\n"
         "4.Search\n5.Find\n6.Exit");


  do
  {
    printf("\n Enter your choice: ");
    scanf("%d",&choice);
    switch (choice)

    {
    case 1:insert();break;
    case 2:ndelete();break;
    case 3:ndisplay();break;
    case 4:search();break;
    case 5:find();break;
    case 6:exit(0);break;
    }
  } while (choice !=6);

  return 0;
}


Node* createNode(int val) {
  Node* new = malloc(sizeof(Node));
  new->data.iValue = val;
  new->linkTo.prev = NULL;
  new->linkTo.next = NULL;

  return new;
}

int getValue() {
  int val;
  printf("\n Enter the data : ");
  scanf("%d",&val);
  printf("%d", val);
  return val;
}

void create()
{
  int val = getValue();

  Node* new = createNode(val);

  if (lhead == NULL) {
    lhead = new;
    ltail = lhead;
  }
  else {
    new->linkTo.prev = ltail;
    ltail->linkTo.next = new;
    ltail = new;
  }

}

Node* traverse(Node* np, int count)
{
  for (; count > 0 && np != NULL; count--) {
    // printf ("count:%d value:%d", count, np->data.iValue);
    np = np->linkTo.next;
  }
  // printf ("Returning np: %p", np);
  return np;

}


```

```
int getPosition() {
  int pos;
  printf("\n Enter the position: ");
  scanf("%d",&pos);
  printf("%d", pos);
  pos = pos-1;
  return pos < 0 ? 0 : pos;  // return a non-negative number
}

void insert()
{
  int pos = getPosition();
  int val = getValue();

  Node* iter = traverse(lhead, pos);
  // printf ("iter: %p ", iter);
  if (iter == NULL) {
    // something is wrong, report it and quit
    printf("\n No such position");

  } else {
    Node* new = createNode(val);

    // insert the new node at the 'iter' location
    // printf ("value at node: %d ", iter->data.iValue);
    Node* locatedNode = iter;
    Node* prevNode = locatedNode->linkTo.prev;

    new->linkTo.next = locatedNode;  // `located` comes after `new`
    locatedNode->linkTo.prev = new;  // and vice-versa

    new->linkTo.prev = prevNode;  // connect with prev
    prevNode->linkTo.next = new;  // and vice-versa

  }

}

void ndelete()
{
  int pos = getPosition();

  Node *nodeToDelete, *prevNode, *nextNode;

  Node* iter = traverse(lhead, pos);
  if (iter == NULL) {
    printf("\n No such position");
    return;
  }
  prevNode = iter->linkTo.prev;
  nextNode = iter->linkTo.next;

  if (iter == lhead) {
    if (nextNode != NULL) {
      lhead->data.iValue = nextNode->data.iValue;
      lhead->linkTo.next = nextNode->linkTo.next;
      free(nextNode);
    }
    else {
      free(lhead);
      printf("\nHead is Free! List is empty!\n");
      lhead = NULL;
      ltail = NULL;
    }

  } else {
    nodeToDelete = iter;
    prevNode->linkTo.next = nextNode;
    if (nextNode != NULL) {
      nextNode->linkTo.prev = prevNode;
    }

    printf("\n The deleted data is %d",nodeToDelete->data.iValue);
    free(nodeToDelete);
  }
}


void ndisplay()
{
  Node* iter = lhead;
  // printf ("Inside display...%p\n", iter);

  if (iter == NULL) printf("\nThe List is empty");
  else {
    printf("\n  The list contains : ");
    for (; iter != NULL; iter = iter->linkTo.next)
      printf("\t %d", iter->data.iValue);
  }

}


void find()
{
  int pos = getPosition();

  Node* iter = traverse(lhead, pos);
  if (iter == NULL)
  {
    printf("\n No such position");
  }
  else {
    Node* prevNode = iter->linkTo.prev;
    Node* nextNode = iter->linkTo.next;

    int iterValue = iter->data.iValue;
    int prevValue = (prevNode != NULL)? prevNode->data.iValue : -1;
    int nextValue = (nextNode != NULL)? nextNode->data.iValue : -1;

    printf("\nThe data around the position are "
           "%d %d %d\n", prevValue, iterValue, nextValue);
  }
}
```

```

void search()  {

}
```