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

//=========================================================

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

node *createNode(void);
void tambahAwal(node **head);
void tambahData(node **head);
void tranverse(node *head);
void insertNode(node **head, node *pPre, node *pNew);
void deleteNode(node **head, node *pPre, node *pCur);
void hapusData(node **head);
void deleteList(node *head);

//=========================================================

void main()
{
  node *head;
  char pilih;

  head = NULL;
  do{
     clrscr();
     printf("masukkan pilihan\n");
     printf("1. tambah data di awal\n");
     printf("2. tambah data di tengah list\n");
     printf("3. hapus data\n");
     printf("4. cetak isi list\n");
     printf("MASUKKAN PILIHAN (tekan q untuk keluar) : ");
     fflush(stdin);
     scanf("%c", &pilih);
     if (pilih == '1')
           tambahAwal(&head);
     else if (pilih == '2')
                 tambahData(&head);
     else if (pilih == '3')
           hapusData(&head);
     else if (pilih == '4'){
           tranverse(head);
         getch();
     }
  } while (pilih != 'q');
  deleteList(head);
}

//=========================================================

node *createNode(void){
  node *ptr;
```

```
  ptr = (node *)malloc(sizeof(node));
  return(ptr);
}

//=========================================================

void tambahAwal(node **head){
  int bil;
  node *pTemp;

  clrscr();
  fflush(stdin);
  printf("masukkan bilangan integer : ");
  fflush(stdin);
  scanf("%d", &bil);
  pTemp = (node *)malloc(sizeof(node));

  if (pTemp != NULL){
           pTemp->data = bil;
      pTemp->next = NULL;
      insertNode(head, NULL, pTemp);
  }
  else{
      printf("Alokasi memori gagal");
      getch();
  }
}

//=========================================================

void tambahData(node **head){
  int pos, bil;
  node *pTemp, *pCur;

  clrscr();
  tranverse(*head);
  printf("\nposisi penyisipan setelah data bernilai : ");
  fflush(stdin);
  scanf("%d", &pos);
  printf("\nData yang disisipkan : ");
  fflush(stdin);
  scanf("%d", &bil);

  pCur = *head;
  while (pCur != NULL && pCur -> data != pos) {
    pCur = pCur -> next;
  }

  pTemp = (node *)malloc(sizeof(node));

  if (pCur == NULL){
     printf("\nnode tidak ditemukan");
     getch();
  }
  else if (pTemp == NULL){
     printf("\nalokasi memeori gagal");
     getch();
  }
```

```
  else{
     pTemp->data = bil;
     pTemp->next = NULL;
     insertNode(head, pCur, pTemp);
  }
}

//=========================================================

void tranverse(node *head){
   //traverse a linked list
      node *pWalker;

   clrscr();
      pWalker = head;
      while (pWalker != NULL){
      printf("%d->", pWalker -> data);
      pWalker = pWalker -> next;
      }
   printf("NULL");
}

//=========================================================

void insertNode(node **head, node *pPre, node *pNew){
    if (pPre == NULL){
       //add before first logical node or to an empty list
          pNew -> next = *head;
       *head = pNew;
    }
    else {
       //add in the middle or at the end
       pNew -> next = pPre -> next;
       pPre -> next = pNew;
   }
}

//=========================================================

void deleteNode(node **head, node *pPre, node *pCur)
{
      if (pPre == NULL)
         *head = pCur -> next;
      else
       pPre -> next = pCur -> next;
      free(pCur);
}

//=========================================================

void hapusData(node **head)
{
  int pos;
  node *pCur, *pPre;

  clrscr();
  tranverse(*head);
  printf("\nData yang akan dihapus : ");
```

```
  fflush(stdin);
  scanf("%d", &pos);

  pPre = NULL;
  pCur = *head;
  //search until the target value is found or the end of the list is
reached
  while (pCur != NULL && pCur -> data != pos) {
    pPre = pCur;
    pCur = pCur -> next;
  }

  if (pCur == NULL){
     printf("\nnode tidak ditemukan");
     getch();
  }
  else
     deleteNode(head, pPre, pCur);
}

//=========================================================

void deleteList(node *head){
  node *pTemp;


  while(head != NULL){
     pTemp = head;
     head = head->next;
     free(pTemp);
  }
}
```