//Program to perform Linked List Operations

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

//Creating a node

struct node { int info;

struct node\* link;

      }\*first=NULL;

void addBeg(int uinfo)

{

  struct node\* ptr = (struct node\*) malloc(sizeof(struct node)); //for dynamically intializing pointer

  if(first==NULL)

    {

      first=ptr;

      ptr->info = uinfo;

      ptr->link= NULL;

    }

  else

    {

      ptr->info = uinfo;

      ptr->link = first;

      first=ptr;

    }

}

void addEnd(int uinfo)

{

if(first==NULL)

    {

      printf("No element present in LL");

      return;

    }

  else

    {

      struct node\* ptr = (struct node\*) malloc(sizeof(struct node)); //for dynamically intializing pointer

      struct node\* cpt = (struct node\*) malloc(sizeof(struct node)); //Availing another new node

      cpt=first;

      while(cpt->link!=NULL) //reaches the last node in LL , ptr will be pointing to last node

      {

        cpt=cpt->link;

      }

      ptr->info = uinfo;

      cpt->link = ptr;

      ptr->link = NULL;

    }

}

void addBtw(int btwinfo, int uinfo)

{

  //btwinfo is the value after which new node has to be inserted

  if(first==NULL)

  {

    printf("No LL is present");

    return;

  }

  else

  {

    struct node\* ptr = (struct node\*) malloc(sizeof(struct node)); //Availing two new nodes

    struct node\* cpt = (struct node\*) malloc(sizeof(struct node));

    cpt=first;

    ptr->info=uinfo;

    while(cpt->info!=btwinfo)

      {

        cpt=cpt->link; //transvering LL till cpt info and given info matches

      }

    ptr->link = cpt->link;

    cpt->link = ptr;

  }

}

void delBeg()

{

  if(first==NULL)

  {

    printf("No element in LL\n");

    return;

  }

  else

  {

    struct node\* ptr = (struct node\*) malloc(sizeof(struct node));

    ptr = first;

    first=ptr->link;

    ptr->link=NULL;

    free(ptr); //deallocation of pointer

    printf("Element Has Been Deleted\n");

  }

}

void delEnd()

{

    if(first==NULL)

    {

      printf("No Element in LL\n");

      return;

    }

    else

    {

      struct node\* ptr = (struct node\*) malloc(sizeof(struct node));

      struct node\* cpt = (struct node\*) malloc(sizeof(struct node));

      ptr=first;

      cpt=first;

      while(ptr->link!=NULL)

      {

        cpt=ptr; //denoting the second last node

        ptr=ptr->link; //tranverses till last node

      }

      cpt->link = NULL;

      ptr->link = NULL;

      free(ptr);

      printf("Element Has Been Deleted\n");

    }

}

void delBtw(int delinfo)

{

  if(first==NULL)

  {

    printf("No Element in LL\n");

    return;

  }

  else

  {

    struct node\* ptr = (struct node\*) malloc(sizeof(struct node));

    struct node\* cpt = (struct node\*) malloc(sizeof(struct node)); //for indicating node before the node to be deleted

    ptr=first;

    while(ptr->info!=delinfo)

    {

      cpt=ptr;

      ptr=ptr->link;

    }

    cpt->link = ptr->link;

    ptr->link = NULL;

    free(ptr);

    printf("Element %d Has Been Deleted \n",delinfo);

  }

}

void printLL()

{

  if(first==NULL)

  {

    printf("No element present inside LL\n");

    return;

  }

  else

  {

  struct node\* ptr = (struct node\*) malloc(sizeof(struct node)); //for dynamically intializing pointer

  ptr=first;

  while(ptr!=NULL)

  {

    printf("%d\t",ptr->info);

    ptr=ptr->link;

  }

  printf("\n");

  }

}

void main()

{

  int uinfo ;

  int btwinfo;

  int delinfo;

  int choice;

  void printMenu()

  {

  printf("Select An Operation:\n");

    printf(" 1. Add Element At Beginning \n");

    printf(" 2. Add Element At End :\n");

    printf(" 3. Add Element in Betweeen :\n");

    printf(" 4. Delete Element From Beginning :\n");

    printf(" 5. Delete Element From Ending :\n");

    printf(" 6. Delete Element In Between:\n");

    printf(" 7. Print Linked List\n");

    printf(" 8. Print Menu\n");

    printf(" 9. Exit \n");

  }

  printMenu();

  while(1)

  {

    printf("Enter Choice:\n");

    scanf("%d" , &choice);

    switch(choice)

    {

      case 1 :  printf("Enter the Element to Add At Beginning\n");

            scanf("%d",&uinfo);

            addBeg(uinfo);

            printf("Element %d has been added\n",uinfo);

            break;

      case 2 :  printf("Enter the element to add at the End\n");

            scanf("%d",&uinfo);

            addEnd(uinfo);

            printf("Element %d has been added\n",uinfo);

            break;

      case 3 :  printf("Enter Where(Element Before Add) You Want To Add New Element\n");

            scanf("%d",&btwinfo);

            printf("Enter the Element to be added\n");

            scanf("%d",&uinfo);

            addBtw(btwinfo,uinfo);

            printf("Element %d has been added\n",uinfo);

            break;

      case 4 :  printf(" Deleting From Beginning\n");

            delBeg();

            break;

      case 5 :  printf(" Deleting From End\n");

            delEnd();

            break;

      case 6 :  printf("Deleting From Between\n");

            printf("Enter The Element To Be Deleted\n");

            scanf("%d",&delinfo);

            delBtw(delinfo);

            break;

      case 7 : printf("Elements in Linked List :\n");

            printLL();

            break;

      case 8 :  printf("Menu :\n");

            printMenu();

            break;

      case 9 :  printf("Exiting \n");

            exit(0);

      default :   printf("Invalid Choice\n , Try Again");

            break;

    }

  }

}

**Output:**





