**Assignment 2**

**Name :-** Dhruv Thakkar

Branch :- CS

Division :- A

Roll no. :- 69

GR Number :- 12111512

**Problem Statement**

WAP to create doubly linked list and perform following

operations on it. A) Insert (all cases) 2. Delete (all cases).

**Solution :**

#include <stdio.h>

#include <stdlib.h>

int size;

struct node \* createDoubleLinkedList(int data);

struct node \* insertLast(struct node \*first, int data);

void traverseLinkedList(struct node \*first);

struct node\* insertFront(struct node \*first, int data);

struct node\* insertAfter(struct node \*first, int y ,int data);

struct node\* deleteLastNode(struct node \*first);

struct node\* deleteFirstNode(struct node \*first);

struct node\* deleteSpecificNode(struct node \*first, int y);

void main(){

   printf("Enter the first node\n");

   int FirstData;

   scanf("%d",&FirstData);

   struct node \*first = createDoubleLinkedList(FirstData);

   while(1){

      printf("Enter 1 for inserting in Front:\n");

      printf("Enter 2 for inserting in Last:\n");

      printf("Enter 3 for inserting After a node:\n");

      printf("Enter 4 for deleting Front node:\n");

      printf("Enter 5 for deleting Last node:\n");

      printf("Enter 6 for deleting a specific node:\n");

      printf("Enter 7 for traversing the linked list\n");

      int choice;

      scanf("%d",&choice);

      if(choice==1){

         int a;

         printf("Enter the element to be inserted\n");

         scanf("%d",&a);

         first = insertFront(first,a);

      }

      else if(choice==2){

         int a;

         printf("Enter the element to be inserted\n");

         scanf("%d",&a);

         first = insertLast(first,a);

      }

      else if(choice==3){

         int a,b;

         printf("Enter the element to be inserted\n");

         scanf("%d",&a);

         printf("Enter the element after which you want to insert\n");

         scanf("%d",&b);

         first = insertAfter(first,b,a);

      }

      else if(choice==4){

         first = deleteFirstNode(first);

         if(first==NULL){

            printf("Enter the first node\n");

            int FirstData;

            scanf("%d",&FirstData);

            struct node \*first = createDoubleLinkedList(FirstData);

            continue;

         }

         printf("First node deleted\n");

      }

      else if(choice==5){

         first = deleteLastNode(first);

         printf("Last node deleted\n");

      }

      else if(choice == 6){

         printf("Enter the element which you want to delete\n");

         int a ;

         scanf("%d",&a);

         first = deleteSpecificNode(first,a);

      }

      else if(choice == 7){

         traverseLinkedList(first);

      }

      else{

         printf("Enter a valid choice!!");

      }

   }

   first = insertLast(first,20);

   first = insertLast(first,30);

   first = insertLast(first,40);

   first = insertFront(first,50);

   first = insertAfter(first,30,70);

   first = deleteLastNode(first);

   first = deleteFirstNode(first);

   first = deleteSpecificNode(first,10);

   traverseLinkedList(first);

}

struct node{

   int data;

   struct node \*next;

   struct node \*prev;

};

struct node \* createDoubleLinkedList(int data){

   struct node \*first = NULL;

   struct node \*new;

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

   new->data=data;

   new->next=NULL;

   new->prev=NULL;

   first = new;

   size=1;

   return first;

}

struct node \* insertLast(struct node \*first, int data){

   if(first==NULL){

      first = createDoubleLinkedList(data);

      return first;

   }

   else{

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

      new->data=data;

      new->next=NULL;

      struct node \*temp=first;

      while(temp->next!=NULL){

      temp=temp->next;

      }

      new->prev=temp;

      temp->next=new;

      size++;

      return first;

   }

}

void traverseLinkedList(struct node \*first){

   struct node \*temp = first;

   //prev->ptr=first;

   printf("Size of Linked list = %d\n",size);

   printf("data = %d Address = %d Prev address = %d Next address = %d\n", first->data ,first ,first->prev ,first->next);

   while(temp->next!=NULL){

      temp=temp->next;

      printf("data = %d Address = %d Prev address = %d Next address = %d\n", temp->data ,temp ,temp->prev ,temp->next);

   }

}

struct node\* insertFront(struct node \*first, int data){

   if(first==NULL){

      first = createDoubleLinkedList(data);

      return first;

   }

   else{

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

      new->prev=NULL;

      new->data=data;

      new->next=first;

      first->prev=new;

      first=new;

      size++;

      return first;

   }

}

struct node\* insertAfter(struct node \*first, int y ,int data){

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

   struct node \*temp = first;

   new->data=data;

   while(temp->data!=y){

      temp=temp->next;

   }

   new->next=temp->next;

   temp->next->prev=new;

   new->prev=temp;

   temp->next=new;

   size++;

   return first;

}

struct node\* deleteLastNode(struct node \*first){

   if(first==NULL){

      printf("List is empty!!");

      return first;

   }

   struct node \*temp=first;

   if(first->next==NULL){

      free(first);

      return NULL;

   }

   while(temp->next!=NULL){

      temp = temp->next;

   }

   temp->prev->next=NULL;

   free(temp);

   size--;

   return first;

}

struct node\* deleteFirstNode(struct node \*first){

   struct node \*temp=first;

   if(first->next==NULL){

      free(first);

      printf("Linked list is now empty\n");

      return NULL;

   }

   temp->next->prev=NULL;

   first=temp->next;

   free(temp);

   size--;

   return first;

}

struct node\* deleteSpecificNode(struct node \*first, int y){

   struct node \*temp = first;

   while(temp->data!=y){

      temp=temp->next;

      if(temp->next==NULL && temp->data !=y){

         printf("Element do not exist!\n");

         return first;

      }

   }

   if(temp==first){

      first = deleteFirstNode(first);

      return first;

   }

   temp->prev->next=temp->next;

   free(temp);

   size--;

   return first;

}

**Output:-**

****

