**Assignment 2**

**Question :-**

1. Write a menu-driven program, which performs the following on a single linked list. (Write a function for each of the operation)

a) Create a linked list.

b) Print the content of the list.

c) Insert a node after the kth node(k may be any interge).

d) Insert a node after the node (first from the start) containing e) Insert a node before the kth node(k may be any interge).

a given value.

f) Insert a node before the node (first from the start) containing a given value.

g) Delete the kth node(k may be any interge). h) Delete the node(first from the start) containing a specified value.

i) Find if two lists are equal(Boolean output)

j) Find the reverse of a list(not just printing in reverse)

k) Concatenate two lists

l) Merge two lists, those are in ascending order(before and after merging).

**Program :-**

#include<stdio.h>

#include<stdlib.h>

struct node

{

   int data;

   struct node \*next;

};

struct node \*create(int size){

    struct node \*head = NULL,\*temp=NULL;

    int i,n;

    printf("Enter the values..\n");

    for(i=0;i<size;i++){

        scanf("%d",&n);

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

        newnode->data = n;

        if(head == NULL){

            head = newnode;

        }

        else{

            temp->next = newnode;

        }

        temp = newnode;

        temp->next = NULL;

    }

    return head;

}

void insert\_before\_ind(struct node \*\*list){

    struct node \*temp = \*list;

    int pos,value;

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

    scanf("%d",&value);

    printf("Enter the position...\n");

    scanf("%d",&pos);

    int i = 0;

    if(pos == 1){

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

        newnode->data = value;

        newnode->next = temp;

        \*list = newnode;

    }

    else{

        while(i < pos - 2){

            temp = temp->next;

            i++;

        }

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

        newnode->data = value;

        newnode->next = temp->next;

        temp->next = newnode;

    }

}

void insert\_after\_ind(struct node \*\*list){

    struct node \*temp = \*list;

    int pos,value;

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

    scanf("%d",&value);

    printf("Enter the position...\n");

    scanf("%d",&pos);

    int i = 0;

    while(i < pos - 1){

        temp = temp->next;

        i++;

    }

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

    newnode->data = value;

    newnode->next = temp->next;

    temp->next = newnode;

}

void insert\_before\_value(struct node \*\*list){

    struct node \*temp = \*list;

    int pos,value;

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

    scanf("%d",&value);

    printf("Enter the positioned value...\n");

    scanf("%d",&pos);

    int i = 0;

    if(pos == temp->data){

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

        newnode->data = value;

        newnode->next = temp;

        \*list = newnode;

    }

    else{

        while(1){

            if(temp->next->data == pos)

                break;

            temp = temp->next;

        }

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

        newnode->data = value;

        newnode->next = temp->next;

        temp->next = newnode;

    }

}

void insert\_after\_value(struct node \*\*list){

    struct node \*temp = \*list;

    int pos,value;

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

    scanf("%d",&value);

    printf("Enter the positioned value...\n");

    scanf("%d",&pos);

    int i = 0;

    while(1){

        if(temp->data == pos)

            break;

        temp = temp->next;

    }

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

    newnode->data = value;

    newnode->next = temp->next;

    temp->next = newnode;

}

struct node \*reverse(struct node \*head){

    struct node \*cur = head;

    struct node \*prev = NULL;

    while(cur != NULL){

        struct node \*temp = cur->next;

        cur->next = prev;

        prev = cur;

        cur = temp;

    }

    return prev;

}

void delete(struct node \*\*head){

    struct node \*temp = \*head;

    int i,pos;

    printf("Enter the position to be deleted...\n");

    scanf("%d",&pos);

    i = 0;

    if(pos == 1){

        \*head = temp->next;

    }

    else{

        while(i < pos - 2){

            temp = temp->next;

            i++;

        }

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

    }

}

void delete\_with\_value(struct node \*\*head){

    struct node \*temp = \*head;

    int value;

    printf("Enter the value to be deleted...\n");

    scanf("%d",&value);

    if(temp->data == value){

        \*head = temp->next;

    }

    else{

        while(temp->next->data != value){

            temp = temp->next;

        }

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

    }

}

void compare(struct node \*list1,struct node \*list2){

    struct node \*temp1 = list1;

    struct node \*temp2 = list2;

    while(temp1 != NULL && temp2 != NULL){

        if(temp1->data != temp2->data){

            break;

        }

        temp1 = temp1->next;

        temp2 = temp2->next;

    }

    if(temp1 == NULL && temp2 == NULL){

        printf("two lists are equal\n");

    }

    else{

        printf("two lists are not equal\n");

    }

}

void concatenate(struct node \*list1,struct node \*list2){

    struct node \*temp = list1;

    while(temp->next){

        temp = temp->next;

    }

    temp->next = list2;

}

void display(struct node \*list){

    struct node \*temp = list;

    while(temp != NULL){

        printf("->%d",temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int count(struct node \*list){

    struct node \*temp = list;

    int c = 0;

    while(temp != NULL){

        temp= temp->next;

        c++;

    }

    return c;

}

struct node \*sort(struct node \*head){

    int i,j,n = count(head);

    printf("%d\n",n);

    for(i = 0;i < n - 1; i++){

        struct node \*temp = head;

        for(j = 0;j < n - i - 1; j++){

            struct node \*p1 = temp;

            struct node \*p2 = temp->next;

            if(p1->data > p2->data){

                int tmp = p1->data;

                p1->data = p2->data;

                p2->data = tmp;

            }

            temp = temp->next;

        }

    }

    return head;

}

struct node \*merge(struct node \*list1,struct node \*list2){

    struct node \*head = NULL,\*temp = NULL;

    struct node \*temp1 = sort(list1);

    struct node \*temp2 = sort(list2);

    while(1){

        if(temp1 == NULL || temp2 == NULL)

            break;

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

        if(temp1->data <= temp2->data){

            newnode->data = temp1->data;

            temp1 = temp1->next;

        }

        else{

            newnode->data = temp2->data;

            temp2 = temp2->next;

        }

        if(head == NULL)

            head = newnode;

        else

            temp->next = newnode;

        temp = newnode;

        temp->next = NULL;

    }

    if(temp1 != NULL){

        while(temp1 != NULL){

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

            newnode->data = temp1->data;

            temp1 = temp1->next;

            temp->next = newnode;

            temp = newnode;

            temp->next = NULL;

        }

    }else{

        while(temp2 != NULL){

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

            newnode->data = temp2->data;

            temp2 = temp2->next;

            temp->next = newnode;

            temp = newnode;

            temp->next = NULL;

        }

    }

    return head;

}

void main(){

    int n;

    struct node \*list1,\*list2;

    printf("1 for creating the Linked List\n2 for display\n3 to insert before a index\n4 to insert after a index\n5 to insert before a value\n6 to insert after a value\n7 to delete a particular position\n8 to delete a value\n9 to compare two lists\n10 to reverse the list\n11 to concatenate two strings\n12 merge two list in sorted order\n");

    while(1){

        int op;

        printf("Enter the operation...\n");

        scanf("%d",&op);

        switch (op)

        {

            case 1:

                printf("Enter the size of the linked list\n");

                scanf("%d",&n);

                list1 = create(n);

                break;

            case 2:

                display(list1);

                break;

            case 3:

                insert\_before\_ind(&list1);

                break;

            case 4:

                insert\_after\_ind(&list1);

                break;

            case 5:

                insert\_before\_value(&list1);

                break;

            case 6:

                insert\_after\_value(&list1);

                break;

            case 7:

                delete(&list1);

                printf("Element deleted...\n");

                break;

            case 8:

                delete\_with\_value(&list1);

                printf("Element deleted...\n");

                break;

            case 9:

                printf("Enter the size of the 2nd list\n");

                scanf("%d",&n);

                list2 = create(n);

                compare(list1,list2);

                break;

            case 10:

                list1 = reverse(list1);

                printf("List reversed...\n");

                break;

            case 11:

                printf("Enter the size of the 2nd list\n");

                scanf("%d",&n);

                list2 = create(n);

                concatenate(list1,list2);

                printf("Two lists concatenated...\n");

                break;

            case 12:

                printf("Enter the size of the 2nd list\n");

                scanf("%d",&n);

                list2 = create(n);

                struct node \*newList = merge(list1,list2);

                printf("The merged list is...\n");

                display(newList);

                break;

            default:

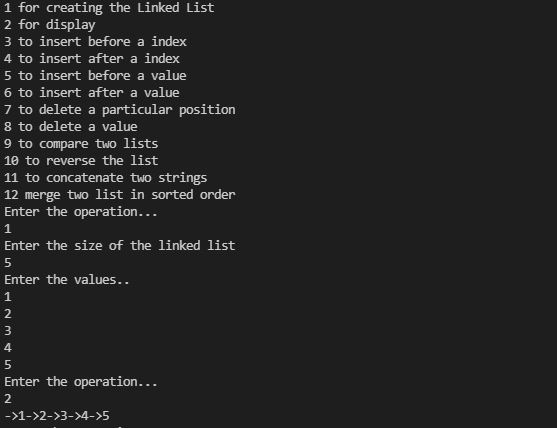
                exit(0);

        }

    }

}

**Output :-**

****

