VISHWAKARMA INSTITUTE OF TECHNOLOGY

NAME	Arpit Sudhir Vidhale
ROLL NO.	60
DIVISION	CS-D
BATCH	B3
PRN NO.	12111229

DS LAB ASSIGNMENT 2

Question:
Consider a CopyList() function that takes a list and returns a complete copy of that list. One pointer can iterate over the original list in the usual way. Two other pointers can keep track of the new list: one head pointer, and one tail pointer which always points to the last node in the new list.

Code:

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

struct Node
{
    int data;
    struct Node *next;
};

void printList(struct Node *head);
void push(struct Node **head, int data);
struct Node *copyList(struct Node *head);

int main(void)
{
    int n;
    printf("enter the no. of elements in a linked list: ");
    scanf("%d", &n);
    printf("enter the elements in a linked list: ");
    int keys[n];
```

```
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &keys[i]);
    }

    struct Node *head = NULL;
    for (int i = n - 1; i >= 0; i--)
    {
        push(&head, keys[i]);
    }
    printf("copied list:");
    struct Node *dup = copyList(head);
    printList(dup);

    return 0;
}

void printList(struct Node *head)
{
    struct Node *ptr = head;
    while (ptr)
    {
        printf("%d —> ", ptr->data);
        ptr = ptr->next;
    }

    printf("NULL");
}

void push(struct Node **head, int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct
    Node)); newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

struct Node *copyList(struct Node *head)
{
    struct Node *current = head;
    struct Node *newList = NULL;
    struct Node *tail = NULL;
```

```
    while (current != NULL)
    {
        if (newList == NULL)
        {
            newList = (struct Node *)malloc(sizeof(struct Node));
            newList->data = current->data;
            newList->next = NULL;
            tail = newList;
        }
        else
        {
            tail->next = (struct Node *)malloc(sizeof(struct Node));
            tail = tail->next;
            tail->data = current->data;
            tail->next = NULL;
        }
        current = current->next;
    }

    return newList;
}
```

Output:

Question:

Program to create, insert, delete and display operations on singly linked list

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
```

```
} *first = NULL;

void create(int A[], int n)
{
    int i;
    struct Node *t, *last;
    first = (struct Node *)malloc(sizeof(struct Node));
    first->data = A[0];
    first->next = NULL;
    last = first;

    for (i = 1; i < n; i++)
    {
        t = (struct Node *)malloc(sizeof(struct Node));
        t->data = A[i];
        t->next = NULL;
        last->next = t;
        last = t;
    }
}
void Display(struct Node *p)
{
    while (p != NULL)
    {
        printf("%d ", p->data);
        p = p->next;
    }
}
void Insert(struct Node *p, int index)
{
    struct Node *t;
    int i;
    int x;
    printf("enter the value to be added in
    list:"); scanf("%d",&x);
    if (index < 0)
        return;
    t = (struct Node *)malloc(sizeof(struct Node));
    t->data = x;
```

```
    if (index == 0)
    {
        t->next = first;
        first = t;
    }
    else
    {
        for (i = 0; i < index - 1; i++)
            p = p->next;
        t->next = p->next;
        p->next = t;
    }
}
int Delete(struct Node *p, int
index) {
    struct Node *q = NULL;
    int x = -1, i;

    if (index < 1)
        return -1;
    if (index == 1)
    {
        q = first;
        x = first->data;
        first = first->next;
        free(q);
        return x;
    }
    else
    {
        for (i = 0; i < index - 1; i++)
        {
            q = p;
            p = p->next;
        }
        q->next = p->next;
        x = p->data;
        free(p);
        return x;
```

```
    }
}

int main()
{
    int choice;
    int A[] = {10, 20, 30, 40, 50};
    create(A, 5);
    printf("\n 1.DISPLAY\n 2.INSERT\n 3.DELETE\n
    4.EXIT"); do
    {
        printf("\n Enter the Choice:");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
        {
            Display(first);
            break;
        }
        case 2:
        {
            int l;
            printf("enter the index of node:");
            scanf("%d", &l);
            Insert(first, l);
            break;
        }
        case 3:
        {
            int l;
            printf("enter the index of node:");
            scanf("%d", &l);
            Delete(first, l);
            break;
        }
        default:
        {
            printf("\n EXIT POINT ");
            break;
```

```
        }
        }
    }
    while(choice!=4);
    return 0;
}
```

Output: