**Practical-7**

**1.** **Write a program to insert a new node into the linked list. A node can be added into the linked list using three ways: [Write code for all the three ways.]**

**a. At the front of the list**

**b. After a given node.**

**c. At the end of the list.**

Aim:

To implement insertion of node in linked list data structure.

Theory:

We will implement insertion of node in linked list data structure using structure pointers and also use malloc function for memory allocation.

Code:

#include<stdio.h>

#include<stdlib.h>

struct node

{

  int info;

  struct node\* next;

};

struct node\* createnode(int i)

{

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

  if(n==NULL)

  {

    printf("\nMemory is full");//OverflowCondition

    exit(0);

  }

  n->info=i;

  return n;

}

struct node \* insertAtEnd(int i,struct node \* start)

{

  struct node\* temp;

  temp=start;

  struct node \*n=createnode(i);

  n->info=i;

  while(temp->next!=NULL)

  {

    temp=temp->next;

  }

  temp->next=n;

  n->next=NULL;

  return start;

}

struct node \* insertAtBeginning(int i,struct node \* start)

{

  struct node \*n=createnode(i);

  n->info=i;

  n->next=start;

  start=n;

  return start;

}

struct node \* insertInBetween(int i,int loc,struct node \* start)

{

  if(loc==1)

  {

    start=insertAtBeginning(i,start);

    return start;

  }

  struct node \*temp;

  temp=start;

  while(loc>=3)

  {

    temp=temp->next;

    if(temp==NULL)

    {

      start=insertAtEnd(i,start);

      return start;

    }

    loc--;

  }

  struct node \*n=createnode(i);

  n->info=i;

  n->next=temp->next;

  temp->next=n;

  return start;

}

int display(struct node \* start)

{

  struct node\* temp;

  temp=start;

  while(temp!=NULL)

  {

    printf("%d-->",temp->info);

    temp=temp->next;

  }

  printf("NULL");

  return 0;

}

int main()

{

  struct node \* start=NULL;

  while(1)

  {

    struct node \* n;

    int a,b;

    printf("\nEnter the number for following choices= \n1.Insert a node at end \n2.Insert a node at beginning \n3.Insert a node in between \n4.Display Linked list \n5.To exit\n");

    scanf("%d",&a);

    switch(a)

    {

      case 1:

      printf("\nEnter the number to be stored=");

      scanf("%d",&b);

        if(start==NULL)

        {

          printf("\nCreating first node");

          struct node \*n=createnode(b);

          start=n;

          n->next=NULL;

        }

        else

        {

          start=insertAtEnd(b,start);

        }

        break;

      case 2:

        printf("\nEnter the number to be stored=");

        scanf("%d",&b);

        if(start==NULL)

        {

          printf("\nCreating first node");

          struct node \*n=createnode(b);

          start=n;

          n->next=NULL;

        }

        else

        {

          start=insertAtBeginning(b,start);

        }

        break;

      case 3:

        printf("\nEnter the number to be stored=");

        scanf("%d",&b);

        int pos;

        printf("\nEnter the location at which you want to insert starting from 1=");

        scanf("%d",&pos);

        if(start==NULL)

        {

          printf("\nCreating first node");

          struct node \*n=createnode(b);

          start=n;

          n->next=NULL;

        }

        else

        {

          start=insertInBetween(b,pos,start);

        }

        break;

      case 4:

        display(start);

        break;

      case 5:

        exit(0);

        break;

      default:

        printf("Invalid choice");

    }

  }

}

Output:

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

**2.** **Write a program to delete a node from the linked list. A node can be deleted from the linked list using three ways: [Write code for all the three ways.]**

**a. Delete from beginning.**

**b. Delete from end.**

**c. Delete from the middle.**

Aim:

To implement deletion of node in linked list data structure.

Theory:

We will implement deletion of node in linked list data structure using structure pointers and also use malloc function for memory allocation.

Code:

#include<stdio.h>

#include<stdlib.h>

struct node

{

  int info;

  struct node\* next;

};

struct node\* createnode(int i)

{

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

  if(n==NULL)

  {

    printf("\nMemory is full");//OverflowCondition

    exit(0);

  }

  n->info=i;

  n->next=NULL;

  return n;

}

struct node \* deleteFromEnd(struct node \* start)

{

  struct node \*temp;

  temp=start;

  if(start==NULL)

  {

    printf("\nThe Linked list is empty");

    return start;

  }

  if(temp->next==NULL)

  {

    start=NULL;

    return start;

  }

  while((temp->next)->next!=NULL)

  {

    temp=temp->next;

  }

  temp->next=NULL;

  return start;

}

struct node \* deleteFromBeginning(struct node \* start)

{

  struct node \* temp;

  temp=start;

  start=temp->next;

  return start;

}

struct node \* deleteFromMiddle(int val,struct node \* start)

{

  if(start->info==val)

  {

    start=deleteFromBeginning(start);

    return start;

  }

  struct node \* prev,\*forward;

  forward=start;

  prev=start;

  while(forward->info!=val)

  {

    prev=forward;

    forward=forward->next;

  }

  prev->next=forward->next;

  return start;

}

int display(struct node \* start)

{

  struct node\* temp;

  temp=start;

  while(temp!=NULL)

  {

    printf("%d-->",temp->info);

    temp=temp->next;

  }

  printf("NULL");

  return 0;

}

struct node \* insertValue(int i,struct node \* start)

{

  struct node\* temp;

  temp=start;

  struct node \*n=createnode(i);

  n->info=i;

  while(temp->next!=NULL)

  {

    temp=temp->next;

  }

  temp->next=n;

  n->next=NULL;

  return start;

}

int main()

{

  struct node \* start=NULL;

  while(1)

  {

    struct node \* n;

    int a,b;

    printf("Enter the number for following choices= \n1.Delete from end \n2.Delete from beginning \n3.Delete from middle \n4.Display Linked list \n5.Enter values for linked list \n6.To exit\n");

    scanf("%d",&a);

    switch(a)

    {

      case 1:

        if(start==NULL)

          printf("\nLinked list is empty ,enter some value for linked list");

        else

          start=deleteFromEnd(start);

        break;

      case 2:

        if(start==NULL)

          printf("\nLinked list is empty ,enter some value for linked list");

        else

          start=deleteFromBeginning(start);

        break;

      case 3:

        if(start==NULL)

          printf("\nLinked list is empty ,enter some value for linked list");

        else

        {

          printf("\nEnter the value to be removed=");

          scanf("%d",&b);

          start=deleteFromMiddle(b,start);

        }

        break;

      case 4:

        display(start);

        break;

      case 5:

        printf("\nEnter the number to be stored=");

      scanf("%d",&b);

        if(start==NULL)

        {

          printf("\nCreating first node");

          struct node \*n=createnode(b);

          start=n;

          n->next=NULL;

        }

        else

        {

          start=insertValue(b,start);

        }

        break;

      case 6:

        exit(0);

        break;

      default:

        printf("Invalid choice");

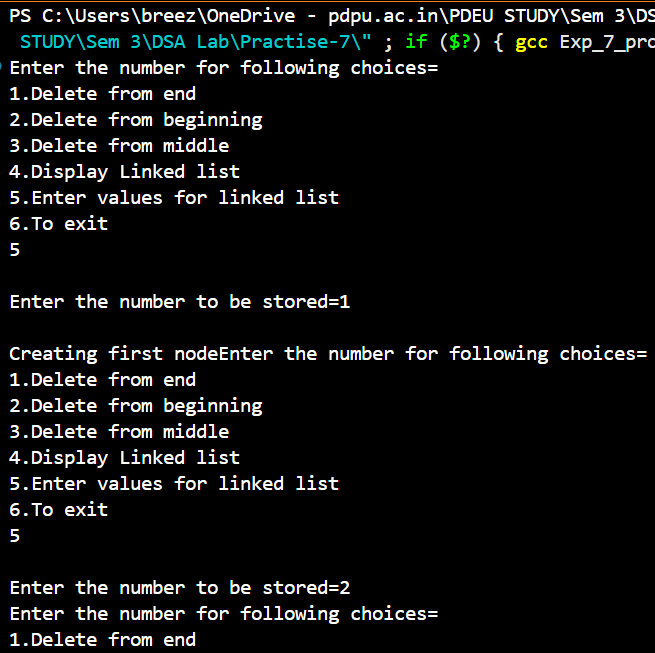
        exit(0);

    }

  }

}

Output:

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.**

**Link for all codes:**

<https://github.com/PanavPatel06/DSA-Lab/tree/main/Practise-7>