NAME: TANISHA.C.A

ROLL NO: 230701390

EX-1: Implementation of Single Linked List

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

struct node
{
   int data;     struct
node *link;
}*first=NULL;

void insert_beg(int);
void insert_end(int);
void insert_mid(int,int);
void del_first(); void
del_last(); void
del_anypos(int); void
display(); void del_all();
void isLast(int); void
isEmpty(); void
findnext(int); void
findprev(int); int
count();
```

```
void search(int);

void insert_beg(int roll)
{
   struct node *newnode;     newnode=(struct
node*)malloc(sizeof(struct node));     newnode-
>data=roll;     if(first==NULL){        newnode->link=NULL;
first=newnode;
   }
   else
   {
      newnode->link=first;
first=newnode;
   }
   printf("Data inserted\n");
}


void insert_end(int roll)
{
   struct node *newnode,*temp;     temp=first;
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=roll;     if(first==NULL)
   {
      newnode->link=NULL;
first=newnode;
   }
   else
   {
      while(temp->link!=0)
      {
```

```
        temp=temp->link;
      }
        newnode->link=NULL;          temp-
>link=newnode;          temp=NULL;
   }
   printf("Data Inserted\n");
}

void display()
{
   struct node *temp=NULL;
temp=first;    if(temp!=NULL){
while(temp!=NULL)
   {
      printf("%d ",temp->data);       temp=temp-
>link;
   }
}
else{    printf("\nNo data
inside");
}
}

void insert_mid(int loc,int roll)
{
   struct node *newnode,*temp=NULL;
   temp=first;
int i=1;
   newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=roll;    int t=count();    if(loc==0)
insert_beg(roll);     else if(loc<t)
```

```
    {
        while(i<loc)
        {
            temp=temp->link;
            i++;

        }
    newnode->link=temp->link;
temp->link=newnode;         printf("Data
Inserted\n");
    }
    else if(loc==t){
insert_end(roll);
    }
    else if(loc>t+1){
printf("Out of bounds");
    }
}

int count(){    struct node
*temp=first;    int count=0;
while(temp!=NULL){
temp=temp->link;
count++;
  }
  return count;
}

void del_first()
{
```

```
   struct node *temp=NULL;
temp=first;     if(first==NULL){
printf("INVALID OPERATION");
   }
   else{        first=temp-
>link;        free(temp);
temp=NULL;
   }
   printf("Data deleted\n");
}


void del_last()
{
   struct node *temp=NULL,*temp1=NULL;
   temp=first;     while(temp-
>link!=0){        temp1=temp;
temp=temp->link;
   }
   free(temp);
   temp=NULL;     temp1-
>link=NULL;     printf("Data
Deleted\n");
}


void del_anypos(int pos)
{
   struct node *temp=NULL,*temp1=NULL;
   temp=first;
if(pos==0)
```

```
   {
      del_first();
   }
   else{        for(int
i=1;i<=pos;i++)
      {
         if(temp==NULL)
{printf("INVALID");
break;}           else{
             temp1=temp;
temp=temp->link;
         }
      }
      if(temp->link!=NULL){        temp1-
>link=(temp->link)->link;}
else{temp1->link=(temp->link);}
free(temp);        temp=NULL;
      temp1=NULL;
     }
     printf("Data Deleted\n");
}

void del_all()
{
   struct node *temp=first,*temp1=NULL;
while(temp!=NULL){        temp1=temp;
temp=temp->link;        free(temp1);
first=NULL;
   }
   temp=NULL;temp1=NULL;
```

```
    printf("\nAll data deleted successfully");
}

void isEmpty()
{
   if(first==NULL){
printf("\nThe list is empty\n");
   }
   else{        printf("\nThe list is not
empty\n");
   }
}

void isLast(int pos)
{
   struct node *temp=first;
   int i=1;
   while(i<pos)
   {
      temp=temp->link;
      i++;
   }
if(temp->link == NULL)
printf("\nIt is the last node");  else
printf("\nIt is not the last node");
}

void search(int data)
{
```

```
   int c=1;     struct node
*temp=first;     if(first==NULL){
printf("\nThe list is empty\n");
   }
   else{
   while(temp!=NULL && temp->data!=data){
temp=temp->link;
      c++;        if(c>count())
printf("No data in list\n");
else     continue;
   }
   printf("\n%d is the position of data\n",c);
}
}

void findnext(int data)
{
   int c=1;     struct node
*temp=first;     if(first==NULL){
printf("\nThe list is empty\n");
   }
   else{
   while(temp!=NULL && temp->data!=data){
temp=temp->link;
      c++;        if(c>count())
printf("No data in list\n");
else     continue;
   }
   printf("\n%d is the position of data\n",c+1);
}
}
```

```
void findprev(int data)
{       int
c=1;
   struct node *temp=first;
if(first==NULL){        printf("\nThe
list is empty\n");
   }
   else{
   while(temp!=NULL && temp->data!=data){
temp=temp->link;
      c++;        if(c>count())
printf("No data in list\n");
else     continue;
   }
   printf("\n%d is the position of data\n",c-1);
}
}

int main()
{
   int n,ch,pos,t;
   printf("MENU DRIVEN PROGRAM:\n");
   printf("0. Exit\n");     printf("1. Insert a node
at the beginning\n");     printf("2. Insert a node
at the end\n");     printf("3. Insert a node after
P\n");     printf("4. Search an element\n");
printf("5. Find next\n");     printf("6. Find
previous\n");     printf("7. isLast\n");
printf("8. isEmpty\n");     printf("9. Delete at
```

```
beg\n");     printf("10. Delete after P\n");
printf("11. Delete at end\n");     printf("12.
Delete list\n");     printf("13. Display\n");
   while(1){ printf("\nEnter your
   choice :  "); scanf("%d",&ch);

   switch (ch)
   {
   case 1:
   printf("\nEnter roll to insert at beginning : ");
scanf("%d",&n);     insert_beg(n);     break;

   case 2:
   printf("\nEnter roll to insert at end : ");
scanf("%d",&n);     insert_end(n);
break;

   case 3:
   printf("Enter P : ");
scanf("%d",&pos);     printf("\nEnter roll
to insert after P : ");     scanf("%d",&n);
insert_mid(pos,n);     break;

   case 4:
   printf("\nEnter data to search : ");
scanf("%d",&n);     search(n);
break;

   case 5:
   printf("\nEnter data to findnext : ");
   scanf("%d",&n);
```

```
   findnext(n);
   break;

   case 6:
   printf("\nEnter data to findprev : ");
scanf("%d",&n);    findprev(n);
break;

   case 7:
   printf("\nEnter position to check last : ");
scanf("%d",&pos);    isLast(pos);    break;

   case 8:
isEmpty();    break;

   case 9:
del_first();    break;

   case 10:
   printf("\nEnter pos to del after P : ");
scanf("%d",&pos);    del_anypos(pos);
break;

   case 11: del_last();
   break;

   case 12:
del_all();    break;
```

```
   case 13:
display();    break;

   default:
      printf("\nMENU EXITED");
      break;
   }
   if(ch==0){
break;
   }
   else
continue;
   }
}
```

```
Enter your choice : 1
Enter the position : 0
Enter the element : 10
Enter your choice : 4
The elements are : 10
Enter your choice : 1
Enter the position : 0
Enter the element : 20
```



```
Enter your choice : 4
The elements are : 20 10
Enter your choice : 1
Enter the position : 1
Enter the element : 25
Enter your choice : 4
```

The elements are : 20 25 10

Enter your choice : 2

Enter the position : 1

Enter your choice : 4

The elements are : 20 10

Enter your choice : 3

Enter the element : 10

Successful. Element 10 is at location 1

Enter your choice : 3 Enter

the element : 25

Unsuccessful.

Enter your choice : 5

Enter your choice : 4

The elements are : 10 20

Enter your choice : 6

.