Practical – 6
Singly Linked List Operations and Applications

A **singly linked list** is a type of linked list that is *unidirectional*, that is, it can be traversed in only one direction from head to the last node (tail).
Each element in a linked list is called a **node**. A single node contains *data* and a pointer to the *next* node which helps in maintaining the structure of the list.

Singly Linked list operations

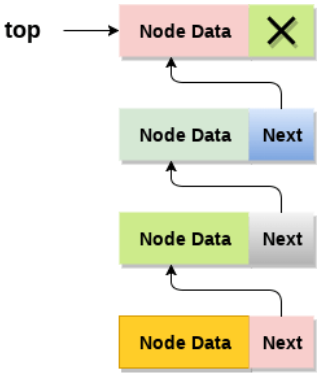
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   char data;
   struct node* next;
}*head=NULL, *temp, *newnode;
void create();
void insertFirst();
void insertEnd();
void insertSpecific(int loc);
void deleteFirst();
void deleteSpecific(int loc);
void display();
int count(struct node *q);
int main()
{
   int choice,loc;
   while(1)
   {
         printf("\n1.Insert at first \n2.Insert at end \n3.Insert at
desired location \n4.Delete at first \n5.Delete at end \n6.Delete at
desired location \n7.Display the list \n8.Exit \nEnter Choice= ");
         scanf("%d",&choice);
         printf("\n");
         switch(choice)
         {
               case 1: insertFirst();break;
               case 2: insertEnd();break;
               case 3: printf("Enter location= ");
                     scanf("%d",&loc);
                     insertSpecific(loc);
                     break;
               case 4: deleteFirst();break;
               case 5: printf("Enter location= ");
```

```
                    scanf("%d",&loc);
                    deleteSpecific(loc);
                    break;
              case 6: display(); break;
              case 7: exit(0); break;
              default: printf("Invalid choice");break;
        }
   }
   return 0;
}
void create()
{
   newnode=(struct node *)malloc(sizeof(struct node));
   if(newnode==NULL)
   {
         printf("No enough memory available\n");
         exit(0);
   }
   newnode->next=NULL;
   printf(" Enter Value= ");
   scanf(" %c",&newnode->data);
}
void insertFirst()
{
   create();
   if(head==NULL)
   {
         head=newnode;
         return;
   }
   newnode->next=head;
   head=newnode;
}
void insertEnd()
{  create();
   if(head==NULL)
   {     head=newnode;
         return;
   }
   temp=head;
   while(temp->next!=NULL)
   {     temp=temp->next;
   }
   temp->next=newnode;
}
void insertSpecific(int loc)
{
   int count=2;
```

```
   struct node *pred;
   temp=head;
   if(loc==1)
   {     insertFirst();
   }
   else
   {     while(temp->next!=NULL && count!=loc)
         {
               count++;
               //pred=temp;
               temp=temp->next;
         }
         create();
         newnode->next=temp->next;
         temp->next=newnode;
   }
}
void deleteFirst()
{
   //struct node *pred;
   if(head==NULL)
   {     printf("List is empty\n");
   }
   else
   {
         temp=head;
         head=head->next;
         free(temp);
   }
}
void deleteSpecific(int loc)
{
   int count=1;
   struct node *pred;
   temp=head;
   if(head==NULL)
   {
         printf("List is empty\n");
   }
   else if(loc==1)
   {
         deleteFirst();
   }
   else
   {     while(temp->next!=NULL && count!=loc)
         {     count++;
               pred=temp;
               temp=temp->next;
```

```
            }
            pred->next=temp->next;
            free(temp);
      }
   }
   void display()
   {
      struct node *temp;
      if(head==NULL)
      {     printf("List is empty\n");
      }
      else
      {
            printf("ELements= ");
            temp=head;
            while(temp!=NULL)
            {
                  printf("%c ",temp->data);
                  temp=temp->next;
            }
            printf("\n");
      }
   }
```

Stack implementation using singly linked list



Stack

ALgorithm

push() - Inserting an element into the Stack

Step 1 - Create a newNode with given value.
Step 2 - Check whether stack is Empty (top == NULL)
Step 3 - If it is Empty, then set newNode → next = NULL.
Step 4 - If it is Not Empty, then set newNode → next = top.
Step 5 - Finally, set top = newNode.

```
// C program for linked list implementation of stack
   #include<stdio.h>
   #include<stdlib.h>
```

```
struct node
{
   int data;
   struct node *next;
}*head=NULL,*newnode,*tmp;
void create();
void push();
void pop();
void peep(int pos);
void display();
int main()
{
   char cc;
   int choice,pos,val;
   while(1)
   {
         printf("\n1.Push \n2.Pop \n3.Peep \n4.Change \n5.Display
\n6.Exit \nEnter Choice= ");
         scanf("%d",&choice);
         printf("\n");
         switch(choice)
         {     case 1: push();break;
               case 2: pop();break;
               case 3: printf("Enter Position : ");
                     scanf("%d",&pos);
                     peep(pos);
                     break;
               case 4:     display();break;
               case 5: exit(0);
               default:printf("Invalid choice\n");break;
         };
   }
}
void create()
{  newnode = (struct node *) malloc(sizeof(struct node *));
   if(newnode==NULL)
   {
         printf("Memory Not Avaliable.\n");
   }
   printf("Enter value= ");
   scanf("%d",&newnode->data);
}
void push()
{
   create();
   newnode->next=head;
   head=newnode;
}
```

```
void pop()
{  if(head == NULL)
   {
         printf("STACK IS EMPTY\n");
   }
   else
   {     tmp=head;
         head = tmp->next;
         free(tmp);
   }
}
void peep(int pos)
{
   int cnt = 1;
   tmp=head;
   while(tmp->next!=NULL&&cnt!=pos)
   {     cnt++;
         tmp=tmp->next;
   }
   printf("Value is: %d\n",tmp->data);
}
void display()
{
   tmp = head;
   if(tmp == NULL)
   {
         printf("STACK IS EMPTY\n");
   }
   else
   {     while (tmp!=NULL)
         {     printf("|    %d   | \n",tmp->data);
               printf("|_________|\n");
               tmp = tmp -> next;
         }
         printf("\n");

   }
}
```

Exercise

1. Write program for all operations of singly link list. (store integer value in list)
   - Creation of List
   - Inserting Node – as First Node, as Last Node, at desired location
   - Deleting Node – at First, at Last, Specific Node
   - Display List

1. Write an algorithm and implement program to perform all stack operations using singly linked list. Implement PUSH, POP, PEEP, Change and DISPLAY.