```
/*************************************************************
 C program for the implementation and performing operations on
                    SINGLE LINKED LIST
*************************************************************/
#include<stdio.h>
#include<stdlib.h>
int create();
int add_beg();
int add_end();
int add_after();
int add_before();
int add_pos();xx
int delete();
int display();
int search();

/*************************************************************
Node declaration for Single linked list
*************************************************************/

struct node
     {
     int info;
     struct node *next;
     };

struct node *start=NULL;
int data,item,pos;

int main()
{
     int ch;
     while(1)
     {
     printf("\n\n****Single Linked List****");
     printf("\n1.Create");
     printf("\n2.Insert Node at Beginning");
     printf("\n3.Insert Node at End");
     printf("\n4.Add After Node");
     printf("\n5.Add Before Node");
     printf("\n6.Add at Position");
     printf("\n7.Delete Node");
     printf("\n8.Display List");
     printf("\n9.Search Element");
```

```
        printf("\n\tEnter Your Choice : ");
        scanf("%d",&ch);

                switch(ch)
                {
                        case 1: create();
                                  break;
                        case 2: add_beg();
                                  break;
                        case 3: add_end();
                                  break;
                        case 4: add_after();
                                  break;
                        case 5: add_before();
                                  break;
                        case 6: add_pos();
                                  break;
                        case 7: delete();
                                  break;
                        case 8: display();
                                  break;
                        case 9: search();
                                  break;
                        default:exit(1);
                }
        }
        return 0;
}

/*************************************************************
 Function to create 1st node of single linked list. If Start
pointer points to NULL, then list is not present (yet to
create) and use function add_beg() to create the 1st node.
If start is not NULL then list is present and can not be
created again
*************************************************************/
int create()
{
        if(start!=NULL)
        {
                printf("List is already present, can't create new");
                return 0;
        }
        else
                add_beg();
}
```

```
/*************************************************************
Function definition to add a node in the beginning of list
*************************************************************/
int add_beg()
{
     printf("\nEnter the Element to be inserted: ");
     scanf("%d",&data);
     struct node *temp;
     temp=(struct node *)malloc(sizeof(struct node));
     temp->info=data;
     temp->next=start;
     start=temp;
     return 0;
}
```

```
/*************************************************************
Function definition to add a node in the end of list
*************************************************************/
int add_end()
{
     printf("\nEnter the Element to be inserted: ");
     scanf("%d",&data);
     struct node *new,*temp;
     new=(struct node *)malloc(sizeof(struct node));
     new->info=data;
     temp=start;
     while(temp->next!=NULL)
          temp=temp->next;
     temp->next=new;
     new->next=NULL;
     return 0;
}
```

```
/*************************************************************
Function definition to add a node after a particular node
whose data is known.
*************************************************************
int add_after()
{
     if(start==NULL)
          {
               printf("\nList is empty");
               return 0;
          }
     printf("\nEnter the Element to be inserted: ");
     scanf("%d",&data);
     printf("Enter the Element after which to insert : ");
     scanf("%d",&item);
     struct node *new,*temp;
     temp=start;
     while(temp!=NULL)
     {
          while(temp->info!=item)
          {
               temp=temp->next;
          }
          new=(struct node *)malloc(sizeof(struct node));
          new->info=data;
          new->next=temp->next;
          temp->next=new;
          return 0;
     }
     printf("\n\t%d is not present in the list",item);
     return 0;
}
```

```
/*************************************************************
Function definition to add a node before a particular node
whose data is known.
*************************************************************/
int add_before()
{
     if(start==NULL)
          {
               printf("\nList is empty");
               return 0;
          }

     printf("\nEnter the Element before which to insert : ");
     scanf("%d",&item);
     if(item==start->info)
          {
               add_beg();
          }
     else
          {
          printf("\nEnter the Element to be inserted: ");
          scanf("%d",&data);
          struct node *new,*temp;
          temp=start;
          while(temp->next!=NULL)
          {
               while(temp->next->info!=item)
               {
                    temp=temp->next;
               }
               new=(struct node *)malloc(sizeof(struct node));
               new->info=data;
               new->next=temp->next;
               temp->next=new;
               return 0;
          }
     printf("\n\t%d item is not present in the list",item);
          return 0;
          }
}
```

```
/*************************************************************
Function definition to add a node before a particular node
at a particular position in the list.
*************************************************************/
int add_pos()
{
     struct node *new,*temp;
     int i;
     temp=start;
     printf("\nEnter the Element to be inserted: ");
     scanf("%d",&data);
     printf("\nEnter the position at which to insert: ");
     scanf("%d",&pos);
     for(i=1;i<pos-1 && pos!=0;i++)
          temp=temp->next;
     if(temp==NULL)
          printf("\n\tThere are less than %d elements",pos);
     else
     {
          new=(struct node *)malloc(sizeof(struct node));
          new->info=data;
          if(pos==1)
          {
               new->next=start;
               start = new;
          }
          else
          {
               new->next=temp->next;
               temp->next=new;
          }
     }
     return 0;
}
```

```
/*************************************************************
Function definition to delete a node whose data is known.
*************************************************************/
int delete()
{
     struct node *new,*temp;
     printf("\nEnter the element to be deleted : ");
     scanf("%d",&item);
     if(start==NULL)
     {
          printf("\n\tList is Empty");
          return 0;
     }

     else if(start->info==item)
     {
          new=start;
          start=start->next;
          free(new);
          return 0;
     }
     temp=start;
     while(temp->next!=NULL)
     {
          if(temp->next->info==item)
          {
               new=temp->next;
               temp->next=new->next;
               free(new);
               return 0;
          }
     temp=temp->next;
     }
     printf("\n\t%d item is not present in the list",item);
     return 0;
}
```

```
/*************************************************************
Function definition to display a complete list. It will
display
"Own address of node, data at node, address of next node"
for all the nodes present in the list
*************************************************************/
int display()
{
     struct node *temp;
     if(start==NULL)
     {
          printf("\n\tList is Empty");
          return 0;
     }
     temp=start;
     printf("\nList : \n");
     while(temp!=NULL)
     {
     printf("\tOwn Addr=%d\tData=%d\tNext Addr=%d\n",temp,temp->info,temp->next);
          temp=temp->next;
     }
     printf("\n\n");
}

/*************************************************************
Function definition to seach an item in the list
*************************************************************/
int search()
{
     struct node *temp=start;
     int pos=1;
     printf("\nEnter the Element to search: ");
     scanf("%d",&item);
     while(temp!=NULL)
     {
          if(temp->info==item)
          {
          printf("\n\tItem %d is found at position %d",item,pos);
               return 0;
          }
          temp=temp->next;
          pos++;
     }
     printf("\n\tItem %d is not found in the list",item);
}
```