SINGLE LINKED LIST

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

        int data;

        struct node *next;

}*head;

void begin_insert();

void last_insert();

void random_insert();

void begin_deletion();

void last_deletion();

void random_deletion();

void display();

void search();

void operation();

int main()

{

        printf("*******operations********");

        printf("\n1.begin
insert\n2.last_insert\n3.random_insert\n4.begin_deletion\n5.last_deletion\n6.random_de
letion\n7.display\n8.search\n9.exit\n");

        printf("*******************\n");

        operation();

return 0;}

void operation()

{
```

```
int choice=0;
while(choice!=9)
{
        printf("enter your choice:");
        scanf("%d",&choice);
        switch (choice)
        {
                case 1:
                        begin_insert();
                        break;
           case 2:
                last_insert();
                break;
           case 3:
                random_insert();
                break;
           case 4:
                begin_deletion();
                break;
           case 5:
                last_deletion();
                break;
           case 6:
                random_deletion();
                break;
           case 7:
                display();
                break;
                case 8:
                    search();
                    break;
                case 9:
                    exit(0);
                default:
                    printf("invaild number!!!! try again!!!\n");
                    operation();
            }
        }

}
void begin_insert()
{
        struct node *ptr;
        int item;
        ptr=(struct node*)malloc(sizeof(struct node *));
        if(ptr==NULL)
        {
                printf("over flow\n");
        }
        else
        {
        printf("enter a number to be inserted:");
        scanf("%d",&item);
        ptr->data=item;
        ptr->next=head;
        head=ptr;
        printf("element insertion is completed\n");
```

```
          }
}
void last_insert()
{
          struct node *ptr,*temp;
          int item;
          ptr=(struct node*)malloc(sizeof(struct node*));
          if(ptr==NULL)
          {
             printf("over flow\n");
          }
          else
          {
                    printf("enter a number to be inserted:");
              scanf("%d",&item);
          ptr->data=item;
          if(head==NULL)
          {
                    ptr->next=head;
                  head=ptr;
              }
              else
              {
                    temp=head;
                    while(temp->next!=NULL)
                    {
                              temp=temp->next;
                              }
                              ptr->next=NULL;
                    temp->next=ptr;
                    printf("insertion is completed\n");
          }
     }
}
void random_insert()
{
int item,loc,i;
struct node *ptr,*temp;
ptr=(struct node*)malloc(sizeof(struct node*));
if(ptr==NULL)
     {
     printf("\nover flow");
     }
else
     {
     printf("enter a number to be inserted:");
     scanf("%d",&item);
     ptr->data=item;
     printf("enter location where node has to be inserted:\n");
     scanf("%d",&loc);
     temp=head;
     for(i=1;i<loc;i++)
     {
     temp=temp->next;
      if(temp==NULL)
       {
          printf("can't inserted\n");
          return;
```

```
                }
    }
    ptr->next=temp->next;
    temp->next=ptr;
    printf(" node inserted\n");
}
}
void begin_deletion()
{
        struct node *ptr;
        if (head==NULL)
        {
                printf("list is empty\n");
        }
        else{
                ptr=head;
                head=ptr->next;
                free(ptr);
                printf("first node is deleted\n");
        }
}
void last_deletion()
{
        struct node *ptr,*ptr1;
        if(head==NULL)
        {
                printf("list is empty\n");
        }
        else if(head->next==NULL)
        {
                head=NULL;
                free(head);
                printf("only one node id deleted\n");
        }
        else
        {
                ptr=head;
                while(ptr->next!=NULL)
                {
                        ptr1=ptr;
                        ptr=ptr->next;
                }
                ptr1->next=NULL;
                free(ptr);
                printf("deleted last node from list\n");
        }
}
void random_deletion()
{
        struct node *ptr,*ptr1;
        int loc,i;
        printf("Enter the location of node when you want to perform deletion:");
        scanf("%d",&loc);
        ptr=head;
        for(i=1;i<loc;i++)
        {
                ptr1=ptr;
                ptr=ptr->next;
        if(ptr==NULL)
        {
                printf("can't delete\n");
                return;
        }
   }
   ptr1->next=ptr->next;
   free(ptr);
   printf("deleted node is %d\n",loc);
}
void display()
{
        struct node *temp;
        if(head==NULL)
        printf("List is empty\n");
        else
        {
                printf("Elements in linked list\n");
                temp=head;
                while(temp!=NULL)
                {
                        printf("%d\n",temp->data);
                        temp=temp->next;
                }

        }
}
void search()
{
        struct node *temp;
        int var,c=0;
        printf("Enter the value of var:");
        scanf("%d",&var);
        temp=head;
        while(temp->next!=NULL)
        {
                c++;
                if(temp->data==var)
        {
                printf("Element is found at %d node\n",c);
                break;
        }
                temp=temp->next;
        }
        if(temp->data==var&&temp->next==NULL)
        {
                printf("Element is  found at %d node\n",c+1);
        }
          else if(temp->data!=var)
        {
                printf("element is not found\n");
                }

}
```