```
// Singly Linked List

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

int choice,a,b,position;

void create();
void display();
void search();
void add_begin();
void add_after();
void insert();
int length();
void del();
void update();

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

void create()
{
   struct node *p, *tmp;
   int num, i, n;
   root = (struct node *)malloc(sizeof(struct node));

   printf("\nEnter the number of nodes : ");
   scanf("%d", &n);

// reads data for the node through keyboard

   printf(" Input data for node 1 : ");
   scanf("%d", &num);
   root->data = num;
   root->link = NULL; // links the address field to NULL
   tmp = root;
// Creating n nodes and adding to linked list
   for(i=2; i<=n; i++)
   {
      p = (struct node *)malloc(sizeof(struct node));

      printf(" Input data for node %d : ", i);
      scanf(" %d", &num);

      p->data = num;      // links the num field of p with num
      p->link = NULL; // links the address field of p with NULL

      tmp->link = p; // links previous node i.e. tmp to the p
      tmp = tmp->link;
```

```
    }
}
void display()
{
    struct node *temp;
    temp = root;
    if (temp==NULL)
    {
        printf("Linked List is Empty.\n");
    }
    else
    {
        printf("\n");
        while (temp != NULL)
        {
            printf(" %d ",temp->data);
            printf("[Own: %d , ",temp);
            printf("Next: %d ]\n",temp->link);
            temp = temp->link;
        }
        printf("\n\n");
    }
}
void search()
{
    int search;
    int n = 1;
    printf("\nEnter the data to search: ");
    scanf("%d",&search);
    struct node *temp;
    temp = root;
    if (temp==NULL)
    {
        printf("Linked List is Empty.\n");
    }
    else
    {
        while (temp != NULL)
        {
            if(search==temp->data)
            {
                printf("Found: %d at Node %d\n\n", search, n);
            }
            temp = temp->link;
            n++;
        }
    }
}

void add_begin(struct node** root, int a)
{
    struct node* p = (struct node*)malloc(sizeof(struct node));
    p->data = a;
    p->link = *root;
```

```
    *root = p;
}
void add_after(struct node **root, int a)
{
    struct node *p = (struct node*)malloc(sizeof(struct node));
    p->data = a;
    p->link = NULL;
    if(*root == NULL)
    {
        *root = p;
    }
    else
    {
        struct node *temp = *root;
        while(temp->link != NULL)
        {
            temp = temp->link;
        }
        temp->link = p;
    }
}
void insert(struct node* root,int data,int position)
    {

     struct node *ptr = (struct node*)malloc(sizeof(struct node));
     ptr->data=data;  //Creating a new node
     int i;
     struct node *temp=root;

     for(i=1;i<position-1;i++)  //moving to the (n-1)th position node in the linked list
     {
        temp=temp->link;
     }

     ptr->link=temp->link;  //Make the newly created node point to link node of ptr temp
     temp->link=ptr;   //Make ptr temp point to newly created node in the linked list
}

int length(struct node *root)
{
    int count = 0;
    if (root == NULL)
    {
        printf("Linked List is Empty.\n");
    }
    else
    {
        struct node *p = NULL;
        p = root;
        while (p != NULL)
        {
            count++;
            p = p->link;
        }
```

```
        printf("\nThe Number of elements in the Linked List are: %d\n\n", count);
    }
}

void del(struct node **root, int a)
{
    struct node *current = *root;
    struct node *previous = *root;
    if (*root == NULL)
    {
        printf("Linked List is Empty.\n");
    }
    else if (a == 1)
    {
        *root = current->link;
        free(current);
        current = NULL;
    }
    else
    {
        while(a != 1)
        {
            previous = current;
            current = current->link;
            a--;
        }
        previous->link = current->link;
        free(current);
        current = NULL;
    }
}

void update()
{
    int pos;
    printf("\nEnter the position to update: ");
    scanf("%d",&pos);
    printf("\nEnter a new value for node %d : ", pos);
    scanf("%d", &a);
    struct node *temp;
    temp = root;
    if (temp==NULL)
    {
        printf("Linked List is Empty.\n");
    }
    else
    {
        for (int i = 1; i <= pos; i++)
        {
            if(i==pos)
            {
                temp->data = a;
            }
            temp = temp->link;
```

```
        }
    }
}

int main()
{
    printf("Singly Linked List Operations\n");
    printf("1. Create\n2. Display\n3. Search\n4. Add at the beginning\n5. Add at the end\n6. Insert data in the middle\
n7. Length of Linked List\n8. Delete\n9. Update Value\n10. Exit\n");
    while (choice != 10)
    {
        printf("Enter your Choice: \n");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
        {
            create();
            printf("\nN.B: Press 11 to see all the options.\n\n");
            break;
        }
        case 2:
        {
            display();
            printf("\nN.B: Press 11 to see all the options.\n\n");
            break;
        }
        case 3:
        {
            search();
            printf("\nN.B: Press 11 to see all the options.\n\n");
            break;
        }
        case 4:
        {
            printf("Enter data to insert at the beginning of the List: \n");
            scanf("%d",&b);
            add_begin(&root, b);
            printf("\nN.B: Press 11 to see all the options.\n\n");
            break;
        }
        case 5:
        {
            printf("Enter data to insert at the end of the List: \n");
            scanf("%d",&b);
            add_after(&root, b);
            printf("\nN.B: Press 11 to see all the options.\n\n");
            break;
        }
        case 6:
        {
            printf("Enter the Position where you want to insert the data: ");
            scanf("%d",&position);
            if(position==1)
```

```
        {

       printf("You must press 3 to insert data at the beginning !\n");

       }
       else
          {
             printf("Enter the Data: ");
             scanf("%d", &b);
             insert(root, b, position);
          }
          printf("\nN.B: Press 11 to see all the options.\n\n");
          break;
       }
       case 7:
       {
          length(root);
          printf("\nN.B: Press 11 to see all the options.\n\n");
          break;
       }
       case 8:
       {
          printf("Enter the Position of the element you want to Delete: \n");
          scanf("%d",&position);
          del(&root, position);
          printf("\nN.B: Press 11 to see all the options.\n\n");
          break;
       }
       case 9:
       {
          update();
          printf("\nN.B: Press 11 to see all the options.\n\n");
          break;
       }
       case 10:
       {
          printf("EXIT\n");
          break;
       }
       case 11:
       {
          printf("Single Linked List Operations\n");
          printf("1. Create\n2. Display\n3. Search\n4. Add at the beginning\n5. Add at the end\n6. Insert data in the mi
ddle\n7. Length of Linked List\n8. Delete\n9. Update Value\n10. Exit\n");
          break;
       }
       default:
       {
          printf ("\n\nPlease Enter a Valid Choice.\nChoose between:1/2/3/4/5/6/7/8/9/10");
       }
       }
   }
   return 0;
}
```