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

struct node
{
   int val;
   struct node *next;
   struct node *prev;
};
typedef struct node n;

n* create_node(int);
void add_node();
void insert_at_first();
void insert_at_end();
void insert_at_position();
void delete_node_position();
void sort_list();
void update();
void search();
void display_from_beg();
```

```
void display_in_rev();


n *new, *ptr, *prev;

n *first = NULL, *last = NULL;

int number = 0;


int main()

{

   int ch;


   printf("\n linked list\n");

   printf("1.insert at beginning \n 2.insert at end\n 3.insert at
position\n4.sort linked list\n 5.delete node at position\n
6.updatenodevalue\n7.search element \n8.displaylist from
beginning\n9.display list from end\n10.exit ");


   while (1)

   {


      printf("\n enter your choice:");

      scanf("%d", &ch);

      switch (ch)
```

```
{
case 1 :
   insert_at_first();
   break;
case 2 :
   insert_at_end();
   break;
case 3 :
   insert_at_position();
   break;
case 4 :
   sort_list();
   break;
case 5 :
   delete_node_position();
   break;
case 6 :
   update();
   break;
case 7 :
   search();
```

```
            break;
        case 8 :
            display_from_beg();
            break;
        case 9 :
            display_in_rev();
            break;
        case 10 :
            exit(0);
        case 11 :
            add_node();
            break;
        default:
            printf("\ninvalid choice");
        }
    }
return 0;
}
/*
 *MEMORY ALLOCATED FOR NODE DYNAMICALLY
 */
```

```
n* create_node(int info)
{
   number++;
   new = (n *)malloc(sizeof(n));
   new->val = info;
   new->next = NULL;
   new->prev = NULL;
   return new;
}
/*
 *ADDS NEW NODE
 */
void add_node()
{

   int info;

   printf("\nenter the value you would like to add:");
   scanf("%d", &info);
   new = create_node(info);
```

```
    if (first == last && first == NULL)
    {

        first = last = new;
        first->next = last->next = NULL;
        first->prev = last->prev = NULL;
    }
    else
    {
        last->next = new;
        new->prev = last;
        last = new;
        last->next = first;
        first->prev = last;
    }
}
/*
 *INSERTS ELEMENT AT FIRST
 */
void insert_at_first()
{
```

```
int info;

printf("\nenter the value to be inserted at first:");
scanf("%d",&info);
new = create_node(info);

if (first == last && first == NULL)
{
   printf("\ninitially it is empty linked list later insertion is done");
   first = last = new;
   first->next = last->next = NULL;
   first->prev = last->prev = NULL;
}
else
{
   new->next = first;
   first->prev = new;
   first = new;
   first->prev = last;
   last->next = first;
```

```
        printf("\n the value is inserted at begining");
    }
}
/*
 *INSERTS ELEMNET AT END
 */
void insert_at_end()
{

    int info;

    printf("\nenter the value that has to be inserted at last:");
    scanf("%d", &info);
    new = create_node(info);

    if (first == last && first == NULL)
    {
        printf("\ninitially the list is empty and now new node is inserted
but at first");
        first = last = new;
        first->next = last->next = NULL;
        first->prev = last->prev = NULL;
```

```
    }
    else
    {
        last->next = new;
        new->prev = last;
        last = new;
        first->prev = last;
        last->next = first;
    }
}
/*
 *INSERTS THE ELEMENT AT GIVEN POSITION
 */
void insert_at_position()
{
    int info, pos, len = 0, i;
    n *prevnode;

    printf("\n enter the value that you would like to insert:");
    scanf("%d", &info);
    printf("\n enter the position where you have to enter:");
```

```
   scanf("%d", &pos);
   new = create_node(info);

   if (first == last && first == NULL)
   {
      if (pos == 1)
      {
         first = last = new;
         first->next = last->next = NULL;
         first->prev = last->prev = NULL;
      }
      else
         printf("\n empty linked list you cant insert at that particular
position");
   }
   else
   {
      if (number < pos)
         printf("\n node cant be inserted as position is exceeding the
linkedlist length");

      else
```

```
        {
            for (ptr = first, i = 1;i <= number;i++)
            {
                prevnode = ptr;
                ptr = ptr->next;
                if (i == pos-1)
                {
                    prevnode->next = new;
                    new->prev = prevnode;
                    new->next = ptr;
                    ptr->prev = new;
                    printf("\ninserted at position %d succesfully", pos);
                    break;
                }
            }
        }
    }
}
/*
 *SORTING IS DONE OF ONLY NUMBERS NOT LINKS
 */
```

```
void sort_list()
{
   n *temp;
   int tempval, i, j;

   if (first == last && first == NULL)
      printf("\nlinked list is empty no elements to sort");
   else
   {
      for (ptr = first,i = 0;i < number;ptr = ptr->next,i++)
      {
         for (temp = ptr->next,j=i;j<number;j++)
         {
            if (ptr->val > temp->val)
            {
               tempval = ptr->val;
               ptr->val = temp->val;
               temp->val = tempval;
            }
         }
      }
```

```
        for (ptr = first, i = 0;i < number;ptr = ptr->next,i++)
            printf("\n%d", ptr->val);
    }
}
/*
 *DELETION IS DONE
 */
void delete_node_position()
{
    int pos, count = 0, i;
    n *temp, *prevnode;

    printf("\n enter the position which u wanted to delete:");
    scanf("%d", &pos);

    if (first == last && first == NULL)
        printf("\n empty linked list you cant delete");

    else
    {
        if (number < pos)
```

```
        printf("\n node cant be deleted at position as it is exceeding
the linkedlist length");


    else
    {
       for (ptr = first,i = 1;i <= number;i++)
       {
          prevnode = ptr;
          ptr = ptr->next;
          if (pos == 1)
          {
             number--;
             last->next = prevnode->next;
             ptr->prev = prevnode->prev;
             first = ptr;
             printf("%d is deleted", prevnode->val);
             free(prevnode);
             break;
          }
          else if (i == pos - 1)
          {
             number--;
```

```
                prevnode->next = ptr->next;
                ptr->next->prev = prevnode;
                printf("%d is deleted", ptr->val);
                free(ptr);
                break;
            }
        }
    }
  }
}
/*
 *UPDATION IS DONE FRO GIVEN OLD VAL
 */
void update()
{
   int oldval, newval, i, f = 0;
   printf("\n enter the value old value:");
   scanf("%d", &oldval);
   printf("\n enter the value new value:");
   scanf("%d", &newval);
   if (first == last && first == NULL)
```

```
        printf("\n list is empty no elemnts for updation");
    else
    {
        for (ptr = first, i = 0;i < number;ptr = ptr->next,i++)
        {
            if (ptr->val == oldval)
            {
                ptr->val = newval;
                printf("value is updated to %d", ptr->val);
                f = 1;
            }
        }
        if (f == 0)
            printf("\n no such old value to be get updated");
    }
}
/*
 *SEARCHING USING SINGLE KEY
 */
void search()
{
```

```
int count = 0, key, i, f = 0;

printf("\nenter the value to be searched:");
scanf("%d", &key);

if (first == last && first == NULL)
   printf("\nlist is empty no elemnets in list to search");
else
{
   for (ptr = first,i = 0;i < number;i++,ptr = ptr->next)
   {
      count++;
      if (ptr->val == key)
      {
         printf("\n the value is found at position at %d", count);
         f = 1;
      }
   }
   if (f == 0)
      printf("\n the value is not found in linkedlist");
}
```

```
}
/*
 *DISPLAYING IN BEGINNING
 */
void display_from_beg()
{
   int i;
   if (first == last && first == NULL)
      printf("\nlist is empty no elemnts to print");
   else
   {
      printf("\n%d number of nodes are there", number);
      for (ptr = first, i = 0;i < number;i++,ptr = ptr->next)
         printf("\n %d", ptr->val);
   }
}
/*
 * DISPLAYING IN REVERSE
 */
void display_in_rev()
{
```

```
    int i;
    if (first == last && first == NULL)
        printf("\nlist is empty there are no elments");
    else
    {
        for (ptr = last, i = 0;i < number;i++,ptr = ptr->prev)
        {
            printf("\n%d", ptr->val);
        }
    }
}
```