```
// Doubly Linked List

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

int choice,a,b,position;

struct node {
    int data;
    struct node * prev;
    struct node * next;
};

struct node *first;
struct node *last;


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

void create()
{
    int n, i, num;

    first = NULL;
    last = NULL;

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

    struct node *temp;

    first = (struct node *)malloc(sizeof(struct node));

    printf("Enter data for node 1 : "); // assigning data in the first node
    scanf("%d", &num);

    first->data = num;
    first->prev = NULL;
    first->next = NULL;
    last = first;

// putting data for rest of the nodes
    for(i=2; i<=n; i++)
        {
            temp = (struct node *)malloc(sizeof(struct node));
```

```
            printf("Enter data for node %d : ", i);
            scanf("%d", &num);
            temp->data = num;
            temp->prev = last;    // new node is linking with the previous node
            temp->next = NULL;

            last->next = temp;   // previous node is linking with the new node
            last = temp;            // assign new node as last node
        }

}

void display()
{
    struct node * temp;
    int n = 1;
    if(first == NULL)
    {
        printf("Linked List is empty.");
    }
    else
    {
        temp = first;
        printf("\nData entered on the list are :\n");

        while(temp != NULL)
        {
            printf("Node %d : %d\n[Own: %d , Previous: %d , Next: %d]\n", n, temp->data, temp, temp->prev, temp->n
ext);
            n++;
            temp = temp->next; // current pointer moves to the next node
        }
    }
}

void search()
{
    int search;
    int n = 1;
    printf("\nEnter the data to search: ");
    scanf("%d",&search);
    struct node *temp;
    temp = first;
    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->next;
            n++;
        }
    }
}

void add_begin(struct node *first, int a)
{
    struct node *p = (struct node*)malloc(sizeof(struct node));
    p->data = a;
    p->prev = NULL;
    p->next = first;
    first->prev = p;
    first = p;
}

void add_after(struct node *first, int a)
{
    struct node *p = (struct node*)malloc(sizeof(struct node));
    p->data = a;
    p->prev = last;
    p->next = NULL;
    if(first == NULL)
    {
        first = p;
    }
    else
    {
        struct node *temp = first;
        while(temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = p;
    }
}

void insert(struct node* first, int value, int position)
    {

     struct node *p = (struct node*)malloc(sizeof(struct node));
     struct node *q;
     p->data = value;  //Creating a new node
     int i;
     struct node *temp = first;

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

     p->prev = temp;   // connecting the prev of p to the node where the loop stopped.
     p->next = temp->next;  //Make the newly created node point to next node of p temp
```

```
    temp->next = p;   //Make p temp point to newly created node in the Linked list

    q = p->next;
    q->prev = p;
}

int length(struct node *first)
{
   int count = 0;
   if (first == NULL)
   {
      printf("Linked List is Empty.\n");
   }
   else
   {
      struct node *p = NULL;
      p = first;
      while (p != NULL)
      {
         count++;
         p = p->next;
      }
      printf("\nThe Number of elements in the Linked List are: %d\n\n", count);
   }
}

void del(struct node *first, int position)
{
   struct node *current = first;
   struct node *previous = first;
   if (first == NULL)
   {
      printf("Linked List is Empty.\n");
   }
   else if (position == 1)
   {
      first = current->next;
      current->next->prev = NULL;
      free(current);
      current = NULL;
   }
   else
   {
      for (int i = 1; i < position; i++)
      {
         previous = current;
         current = current->next;
      }
      //previous->prev = current->prev;
      previous->prev = current->prev->prev;
      current->next->prev = previous;
      previous->next = current->next;
      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 = first;
    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->next;
        }
    }
}

int main()
{
    int choice;

 printf("Doubly 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 nexted List\n8. Delete\n9. Update Value\n10. Exit\n");
    while (choice != 10)
    {
        printf("\nEnter your Choice: ");
        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(first, 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(first, 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(first, b, position);
   }
   printf("\nN.B: Press 11 to see all the options.\n\n");
   break;
}
case 7:
{
   length(first);
   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(first, 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("Doubly 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 nexted 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;
}
```