Doubly linked list

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

struct node
{
   struct node *prev;
   int n;
   struct node *next;
}*head,*temp,*temp1,*temp2;

void insatbeg();
void insatend();
void insatpos();
void display();

void search();

void delete();

int count = 0;

void main()
{
   int ch;

   head = NULL;
   temp = temp1 = NULL;

   printf("\n 1 - Insert at beginning");
   printf("\n 2 - Insert at end");
   printf("\n 3 - Insert at position i");
   printf("\n 4 - Delete at i");
   printf("\n 5 - Display from beginning");
   printf("\n 6 - Search for element");


   printf("\n 7 - Exit");
```

```
   while (1)
   {
      printf("\n Enter choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         insatbeg();
         break;
      case 2:
         insatend();
         break;
      case 3:
         insatpos();
         break;
      case 4:
         delete();
         break;
      case 5:
         display();
         break;


      case 6:
         search();
         break;


      case 7:
         exit(0);
      default:
         printf("\n Wrong choice menu");
      }
   }
}

/* TO create an empty node */
void create()
{
```

```
    int data;

    temp =(struct node *)malloc(1*sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter value to node : ");
    scanf("%d", &data);
    temp->n = data;
    count++;
}

/*  TO insert at beginning */
void insatbeg()
{
    if (head == NULL)
    {
        create();
        head = temp;
        temp1 = head;
    }
    else
    {
        create();
        temp->next = head;
        head->prev = temp;
        head = temp;
    }
}

/* To insert at end */
void insatend()
{
    if (head == NULL)
    {
        create();
        head = temp;
        temp1 = head;
    }
    else
```

```
    {
        create();
        temp1->next = temp;
        temp->prev = temp1;
        temp1 = temp;
    }
}

/* To insert at any position */
void insatpos()
{
    int pos, i = 2;

    printf("\n Enter position to be inserted : ");
    scanf("%d", &pos);
    temp2 = head;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Position out of range to insert");
        return;
    }
    if ((head == NULL) && (pos != 1))
    {
        printf("\n Empty list cannot insert other than 1st position");
        return;
    }
    if ((head == NULL) && (pos == 1))
    {
        create();
        head = temp;
        temp1 = head;
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
```

```
            i++;
        }
        create();
        temp->prev = temp2;
        temp->next = temp2->next;
        temp2->next->prev = temp;
        temp2->next = temp;
    }
}

/* To delete an element */
void delete()
{
    int i = 1, pos;

    printf("\n Enter position to be deleted : ");
    scanf("%d", &pos);
    temp2 = head;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Error : Position out of range to delete");
        return;
    }
    if (head == NULL)
    {
        printf("\n Error : Empty list no elements to delete");
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
```

```
        {
            printf("Node deleted from list");
            free(temp2);
            temp2 = head = NULL;
            return;
        }
    }
    if (temp2->next == NULL)
    {
        temp2->prev->next = NULL;
        free(temp2);
        printf("Node deleted from list");
        return;
    }
    temp2->next->prev = temp2->prev;
    if (i != 1)
        temp2->prev->next = temp2->next;    /* Might not need this statement if i == 1 check */
    if (i == 1)
        head = temp2->next;
    printf("\n Node deleted");
    free(temp2);
  }
  count--;
}

/* Traverse from beginning */
void display()
{
  temp2 = head;

  if (temp2 == NULL)
  {
    printf("List empty to display \n");
    return;
  }
  printf("\n Linked list elements from begining : ");

  while (temp2->next != NULL)
  {
```

```
      printf(" %d ", temp2->n);
      temp2 = temp2->next;
   }
   printf(" %d ", temp2->n);
}

/* To traverse from end recursively */

/* To search for an element in the list */
void search()
{
   int data, count = 0;
   temp2 = head;

   if (temp2 == NULL)
   {
      printf("\n Error : List empty to search for data");
      return;
   }
   printf("\n Enter value to search : ");
   scanf("%d", &data);
   while (temp2 != NULL)
   {
      if (temp2->n == data)
      {
         printf("\n Data found in %d position",count + 1);
         return;
      }
      else
          temp2 = temp2->next;
         count++;
   }
   printf("\n Error : %d not found in list", data);
}
```

output

```
1 - Insert at beginning
2 - Insert at end
3 - Insert at position i
4 - Delete at i
5 - Display from beginning
6 - Search for element
7 - Exit
Enter choice : 1

Enter value to node : 3

Enter choice : 1

Enter value to node : 4

Enter choice : 5

Linked list elements from begining :  4  3
Enter choice : 2

Enter value to node : 6

Enter choice : 5

Linked list elements from begining :  4  3  6
Enter choice : 3

Enter position to be inserted : 2

Enter value to node : 1

Enter choice : 5

Linked list elements from begining :  4  1  3  6
Enter choice : 4

Enter position to be deleted : 1

Node deleted
Enter choice : 5

Linked list elements from begining :  1  3  6
```

```
Enter position to be inserted : 2

Enter value to node : 1

Enter choice : 5

Linked list elements from begining :  4  1  3  6
Enter choice : 4

Enter position to be deleted : 1

Node deleted
Enter choice : 5

Linked list elements from begining :  1  3  6
Enter choice : 4

Enter position to be deleted : 3
Node deleted from list
Enter choice : 5

Linked list elements from begining :  1  3
Enter choice : 6

Enter value to search : 3

Data found in 2 position
Enter choice : 6

Enter value to search : 7

Error : 7 not found in list
Enter choice :
```