Doubly Linked List

C Program to delete node from Doubly Linked List

```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node * prev;
   struct node * next;
}*head, *last;
void createList(int n);
void displayList();
void deleteFromBeginning();
void deleteFromEnd();
void deleteFromN(int position);
void main()
{
   int n, data, choice=1;

   head = NULL;
   last = NULL;

  while(choice != 0)
   {

      printf("DOUBLY LINKED LIST PROGRAM\n");
```

```
printf("1. Create List\n");
printf("2. Delete node - from beginning\n");
printf("3. Delete node - from end\n");
printf("4. Delete node - from N\n");
printf("5. Display list\n");
printf("0. Exit\n");
printf("Enter your choice : ");

scanf("%d", &choice);

switch(choice)
{
   case 1:
      printf("Enter the total number of nodes in list: ");
      scanf("%d", &n);
      createList(n);
      break;
   case 2:
      deleteFromBeginning();
      break;
   case 3:
      deleteFromEnd();
      break;
   case 4:
```

```
            printf("Enter the node position which you want to delete: ");
            scanf("%d", &n);
            deleteFromN(n);
            break;
        case 5:
            displayList();
            break;
        case 0:
            break;
        default:
            printf("Error! Invalid choice. Please choose between 0-5");
      }

      printf("\n\n\n\n\n");
   }

getch();
}
void createList(int n)
{
   int i, data;
   struct node *newNode;

   if(n >= 1)
   {
```

```
head = (struct node *)malloc(sizeof(struct node));

printf("Enter data of 1 node: ");
scanf("%d", &data);

head->data = data;
head->prev = NULL;
head->next = NULL;

last = head;
for(i=2; i<=n; i++)
{
   newNode = (struct node *)malloc(sizeof(struct node));

   printf("Enter data of %d node: ", i);
   scanf("%d", &data);

   newNode->data = data;
   newNode->prev = last;
   newNode->next = NULL;

   last->next = newNode;
   last = newNode;        }
```

```
        printf("\nDOUBLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
void deleteFromBeginning()
{
    struct node * toDelete;

    if(head == NULL)
    {
        printf("Unable to delete. List is empty.\n");
    }
    else
    {
        toDelete = head;

        head = head->next;
        if (head != NULL)
            head->prev = NULL;
        free(toDelete);
        printf("SUCCESSFULLY DELETED NODE FROM BEGINNING OF THE LIST.\n");
    }
}
void deleteFromEnd()
{
    struct node * toDelete;
```

```
    if(last == NULL)
    {
        printf("Unable to delete. List is empty.\n");
    }
    else
    {
        toDelete = last;

        last = last->prev;
        if (last != NULL)
            last->next = NULL;
        free(toDelete);
        printf("SUCCESSFULLY DELETED NODE FROM END OF THE LIST.\n");
    }
}
void deleteFromN(int position)
{
    struct node *current;
    int i;

    current = head;
    for(i=1; i<position && current!=NULL; i++)
    {
        current = current->next;
```

```
    }

    if(position == 1)
    {
        deleteFromBeginning();
    }
    else if(current == last)
    {
        deleteFromEnd();
    }
    else if(current != NULL)
    {
        current->prev->next = current->next;
        current->next->prev = current->prev;

        free(current);
        printf("SUCCESSFULLY DELETED NODE FROM %d POSITION.\n", position);
    }
    else
    {
        printf("Invalid position!\n");
    }
}
void displayList()
{
```

```
    struct node * temp;
    int n = 1;

    if(head == NULL)
    {
        printf("List is empty.\n");
    }
    else
    {
        temp = head;
        printf("DATA IN THE LIST:\n");

        while(temp != NULL)
        {
            printf("DATA of %d node = %d\n", n, temp->data);

            n++;
            temp = temp->next;
        }
    }
}
```