

Please select an operation to perform from the below list

1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit

Enter your choice: 1

Enter the number of nodes: 3

Enter the data: 1

2

3

Please select an operation to perform from the below list

1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit

Enter your choice: 4

[1 2 3]

Please select an operation to perform from the below list

1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit

Enter your choice: 3

Select a position from where you to want to delete the element

1. Beginning of the List
2. At the end of the list
3. Somewhere in between
4. Exit the delete operation

Program code:

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

#include <malloc.h>

// Defining Structure
typedef struct nd
{
    int data;
    struct nd *next;
} nd;
nd *createList();
nd *Insert_beg(nd *head, int x);
nd *Insert_end(nd *head, int x);
nd *Insert_mid(nd *head, int x);
nd *Delete_beg(nd *head);
nd *Delete_end(nd *head);
nd *Delete_mid(nd *head);
void PrintList(nd *head);

// Main Function
void main()
{
    int choice, insert_option, delete_option, x;
    nd *head = NULL;
    printf("Welcome to the implementation of the singly linked list ! \n");
    do
    {
        printf("Please select an operation to perform from the below list \n");
        printf(" 1. Create a List \n 2. Insert a nd \n 3. Delete a nd \n 4. Print the existing list \n 5. Exit \n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        printf("\n \n");
        switch (choice)
        {
            case 1:
                head = createList();
                break;
            case 2:
                do
```

```

{
    printf("Select a position where you to want to insert new nd \n");
    printf(" 1. Beginning of the List \n 2. At the end of the list \n 3. Insert in between \n
4. Exit the insert operation \n");
    printf("Enter your choice: ");
    scanf("%d", &insert_option);
    switch (insert_option)
    {
        case 1:
            printf("Enter the data to be inserted: ");
            scanf("%d", &x);
            head = Insert_beg(head, x);
            break;
        case 2:
            printf("Enter the data to be inserted: ");
            scanf("%d", &x);
            head = Insert_end(head, x);
            break;
        case 3:
            printf("Enter the data to be inserted: ");
            scanf("%d", &x);
            head = Insert_mid(head, x);
            break;
        case 4:
            printf("Insert operation Exit");
            break;
        default:
            printf("Please enter a valid choide: 1, 2, 3, 4");
    }
} while (insert_option != 4);
printf("\n \n");
break;
case 3:
do
{
    printf("Select a position from where you to want to delete the element \n");
    printf(" 1. Beginning of the List \n 2. At the end of the list \n 3. Somewhere in
between \n 4. Exit the delete operation \n");
    printf("Enter your choice: ");
    scanf("%d", &delete_option);
    switch (delete_option)
    {
        case 1:
            head = Delete_beg(head);

```

```

        break;
        case 2:
            head = Delete_end(head);
            break;
        case 3:
            head = Delete_mid(head);
            break;
        case 4:
            printf("Delete Operation Exit");
            break;
        default:
            printf("Please enter a valid choide: 1, 2, 3, 4");
    }
} while (delete_option != 4);
printf("\n \n");
break;
case 4:
    PrintList(head);
    break;
case 5:
    printf("Exit: Program Finished !!");
    break;
default:
    printf("Please enter a valid choide: 1, 2, 3, 4, 5");
}
} while (choice != 5);
}

```

```

// Function to create List
nd *createList()
{
    nd *head, *p;
    int i, n;
    head = NULL;
    printf("Enter the number of nds: ");
    scanf("%d", &n);
    printf("Enter the data: ");
    for (i = 0; i <= n - 1; i++)
    {
        if (head == NULL)
        {
            p = head = (nd *)malloc(sizeof(nd));
        }
        else

```

```

    {
        p->next = (nd *)malloc(sizeof(nd));
        p = p->next;
    }
    p->next = NULL;
    scanf("%d", &(p->data));
}
printf("\n \n");
return (head);
}

```

// Function to insert element

nd *Insert_beg(nd *head, int x)

```

{
    nd *p;
    p = (nd *)malloc(sizeof(nd));
    p->data = x;
    p->next = head;
    head = p;
    return (head);
}

```

nd *Insert_end(nd *head, int x)

```

{
    nd *p, *q;
    p = (nd *)malloc(sizeof(nd));
    p->data = x;
    p->next = NULL;
    if (head == NULL)
        return (p);
    for (q = head; q->next != NULL; q = q->next)
        ;
    q->next = p;
    return (head);
}

```

nd *Insert_mid(nd *head, int x)

```

{
    nd *p, *q;
    int y;
    p = (nd *)malloc(sizeof(nd));
    p->data = x;
    p->next = NULL;
    printf("After which element you want to insert the new element ?");
    scanf("%d", &y);
    for (q = head; q != NULL && q->data != y; q = q->next)

```

```

;
if (q != NULL)
{
p->next = q->next;
q->next = p;
}
else
printf("ERROR !! Data Not Found");
return (head);
}

```

// Function to delete element

```
nd *Delete_beg(nd *head)
```

```

{
    nd *p, *q;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    p = head;
    head = head->next;
    free(p);
    return (head);
}

```

```
nd *Delete_end(nd *head)
```

```

{
    nd *p, *q;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    p = head;
    if (head->next == NULL)
    {
        head = NULL;
        free(p);
        return (head);
    }
    for (q = head; q->next->next != NULL; q = q->next)
    p = q->next;
    q->next = NULL;
    free(p);
}

```

```

        return (head);
    }
    nd *Delete_mid(nd *head)
    {
        nd *p, *q;
        int x, i;
        if (head == NULL)
        {
            printf("Empty Linked List");
            return (head);
        }
        printf("Enter the data to be deleted: ");
        scanf("%d", &x);
        if (head->data == x)
        {
            p = head;
            head = head->next;
            free(p);
            return (head);
        }
        for (q = head; q->next->data != x && q->next != NULL; q = q->next)
        if (q->next == NULL)
        {
            printf("ERROR !! Data Not Found");
            return (head);
        }
        p = q->next;
        q->next = q->next->next;
        free(p);
        return (head);
    }
}

```

```

// Function to print the existing list
void PrintList(nd *head)
{
    nd *p;
    printf("[ ");
    for (p = head; p != NULL; p = p->next)
    {
        printf("%d \t", p->data);
    }
    printf(" ]");
    printf("\n \n");
}

```