```c
//  Created by Ritwik Chandra Pandey on 07/02/21.
//  183215
//  UCS-3 Assignment
// Linked List Implementation of List


#include<stdio.h>
#include<stdlib.h>
struct node {
    int data;
    struct node *next;
};
typedef struct node* NODE;

NODE createNode(){
    NODE temp;
    temp = (NODE)calloc(1,sizeof(struct node));
    temp -> next = NULL;
    return temp;
}




NODE addNodes(NODE first, int x) {
    NODE temp, lastNode = first;
    temp = createNode();
    temp -> data = x;
    if (first == NULL) {
        first = temp;
    } else {
        while (lastNode -> next != NULL) {
            lastNode = lastNode -> next;
        }
        lastNode -> next = temp;
```

```
    }
    return first;

}
```

```
int count(NODE first) {
    NODE temp = first;
    int sum = 0;
    while (temp != NULL) {
        sum++;
        temp = temp -> next;
    }
    return sum;
}
```

```
NODE insertAtBegin(NODE first, int x) {
    NODE temp;
    temp = createNode();
    temp -> data = x;
    temp -> next = first;
    first = temp;
    return first;
}
```

```
NODE insertAtEnd(NODE first, int x) {
    NODE temp, lastNode = first;
    temp = createNode();
    temp -> data = x;
    if (first == NULL) {
        first = temp;
```

```
    } else {
        while (lastNode -> next != NULL) {
            lastNode = lastNode -> next;
        }
        lastNode -> next = temp;
    }
    return first;
}



NODE insertAtPosition(NODE first, int pos, int x) {
    NODE temp, prevPos = first, last = first;
    int i;
    for (i = 1; i < pos; i++) {
        if (last == NULL) {
            printf("No such position in SLL."
                   " So insertion is not possible\n");
            return first;
        }
        prevPos = last;
        last = last -> next;
    }
    if (pos <= 0) {
        printf("No such position in SLL."
               " So insertion is not possible\n");
        return first;
    }
    temp = createNode();
    temp -> data = x;
    if (pos == 1) {
        temp -> next = first;
        first = temp;
    } else {
        temp -> next = prevPos -> next;
        prevPos -> next = temp;
```

```
    }
    return first;
}
```

```
void traverseList(NODE first) {
    NODE temp = first;
    while (temp != NULL) {
        printf("%d --> ", temp -> data);
        temp = temp -> next;
    }
    printf("NULL\n");
}
```

```
int searchPosOfEle(NODE first, int key) {
    NODE currentNode = first;
    int count = 0;
    if (currentNode == NULL) {
        return count;
    }
    while (currentNode != NULL &&
           currentNode -> data != key) {
        if (currentNode -> next == NULL) {
            return 0;
        }
        count++;
        currentNode = currentNode -> next;
    }
    return (count + 1);
```

```
}
```

```
NODE deleteAtBegin(NODE first) {
    NODE temp = first;
    first = first -> next;
    printf("The deleted element from SLL : "
           "%d\n", temp -> data);
    free(temp);
    return first;
}
```

```
NODE deleteAtEnd(NODE first) {
    NODE prev=NULL, lastNode = first;
    if (lastNode -> next == NULL) {
        first = first -> next;
    } else {
        while (lastNode -> next != NULL) {
            prev = lastNode;
            lastNode = lastNode -> next;
        }
        prev -> next = NULL;
    }
    printf("The deleted element from SLL : "
           "%d\n", lastNode -> data);
    free(lastNode);
    return first;}
```

```
NODE deleteAtPosition(NODE first, int pos) {
    NODE prevPos = first, lastNode = first;
    int i;
    if (pos == 1) {
        first = first -> next;
    } else {
        for (i = 1; i < pos; i++) {
            if (lastNode == NULL) {
                printf("No such position in SLL."
                       " So deletion is not possible\n");
                return first;
            }
            prevPos = lastNode;
            lastNode = lastNode -> next;
        }
        if (lastNode == NULL || pos <= 0) {
            printf("No such position in SLL."
                   " So deletion is not possible\n");
            return first;
        } else {
            prevPos -> next = lastNode -> next;
        }
    }
    printf("The deleted element from SLL : "
           " %d\n", lastNode -> data);
    free(lastNode);
    return first;
}
NODE deleteList(NODE head_ref)
{

    NODE current = head_ref;
    NODE next;

    while (current != NULL)
    {
        next = current->next;
```

```
        free(current);
        current = next;
    }


    head_ref = NULL;
    return head_ref;
}




int main() {
    NODE first = NULL;
    int select = 0,x,pos;
    printf("\t\tLINKED LIST IMPLEMENTATION OF LIST\n\n");
    do{
        printf("\t1.ADD NODES\n\t2.COUNT\n\t3.INSERT AT BEGIN\n\t4.INSERT AT END\n\t5.INSERT AT
POSITION\n\t6.TRAVERSE LIST\n\t7.SEARCH\n\t8.DELETE AT BEGIN\n\t9.DELETE AT END\n\t10.DELETE AT
POSITION\n\t11.DELETE LIST\n\t12.EXIT\n");


        printf("\tPlease Enter Your Choice\n");
        scanf("%d",&select);
        switch(select)
        {
            case 1:
                printf("Enter elements up to -1 : ");
                scanf("%d", &x);
                while (x != -1) {
                    first = addNodes(first, x);
                    scanf("%d", &x);
                }
                printf("-----------------------------------------------------------\n");
                break;
            case 2:
                printf("The number of nodes in a SLL are : %d\n", count(first));
```

```
    printf("-----------------------------------------------------------\n");
    break;
case 3:
    printf("Enter any number : ");
    scanf("%d", &x);
    first = insertAtBegin(first, x);
    printf("-----------------------------------------------------------\n");
    break;
case 4:
    printf("Enter any number : ");
    scanf("%d", &x);
    first = insertAtEnd(first, x);
    printf("-----------------------------------------------------------\n");
    break;
case 5:

    printf("Enter a position : ");
    scanf("%d", &pos);
    printf("Enter an element : ");
    scanf("%d", &x);
    first = insertAtPosition(first, pos, x);

    printf("-----------------------------------------------------------\n");
    break;
case 6:
    if (first == NULL) {
        printf("Single Linked List is empty\n");
    } else {
        printf("The elements in SLL are : ");
        traverseList(first);
    }

    printf("-----------------------------------------------------------\n");
    break;
case 7:

    printf("Enter search element : ");
```

```
            scanf("%d", &x);
            pos = searchPosOfEle(first, x);
            if (pos == 0) {
                printf("The given element %d is not found in the given SLL\n", x);
            } else {
                printf("The given element %d is found at position : %d\n", x, pos);
            }
            printf("----------------------------------------------------------------\n");
            break;
        case 8:
            if (first == NULL) {
                printf("Single linked list is empty."
                       " So deletion is not possible\n");
            } else {
                first = deleteAtBegin(first);
            }

            printf("----------------------------------------------------------------\n");
            break;
        case 9:
            if (first == NULL) {
                printf("Single linked list is empty."
                       " So deletion is not possible\n");
            } else {
                first = deleteAtEnd(first);
            }
            printf("----------------------------------------------------------------\n");
            break;
        case 10:
            if (first == NULL) {
                printf("Single Linked List is empty."
                       " So deletion is not possible\n");
            } else {
                printf("Enter position : ");
                scanf("%d", &pos);
                first = deleteAtPosition(first, pos);
            }
```

```
                printf("-------------------------------------------------------------\n");
                break;
            case 11:
                first = deleteList(first);
                printf("-------------------------------------------------------------\n");
                break;

            case 12:
                break;

            default:
                printf("\t\n\nYou have not entered the right choice\n\n");

        }
    }while(select!=12);

}
```