```
// Created by Ritwik Chandra Pandey on 14/02/21.
//  183215
// Double Linked List Implementation


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


struct node {
    int data;
    struct node *prev;
    struct node *next;
};
typedef struct node *NODE;
NODE first = NULL;


NODE createNodeInDLL(){
    NODE temp;
    temp = (NODE) malloc(sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    return temp;
}



NODE addNodesInDLL(NODE first, int x) {
    NODE temp, lastNode = first;
    temp = createNodeInDLL();
    temp -> data = x;
    if (first == NULL) {
        first = temp;
    } else {
```

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

NODE insertAtBeginInDLL(NODE first, int x) {
    NODE temp;
    temp = createNodeInDLL();
    temp -> data = x;
    if (first != NULL) {
        temp -> next = first;
        first -> prev = temp;
    }
    first = temp;
    return first;
}
NODE insertAtEndInDLL(NODE first, int x) {
    NODE temp, lastNode = first;
    temp = createNodeInDLL();
    temp -> data = x;
    if (first == NULL) {
        first = temp;
    } else {
        while (lastNode -> next != NULL) {
            lastNode = lastNode -> next;
        }
        lastNode -> next = temp;
        temp -> prev = lastNode;
    }
    return first;
}
NODE insertAtPositionInDLL(NODE first, int position, int x) {
    NODE temp, lastNode = first;
```

```
    int i;
    for (i = 1; i < (position - 1); i++) {
        if (lastNode -> next == NULL) {
            printf("No such position in DLL. So insertion is not possible\n");
            return first;
        }
        lastNode = lastNode -> next;
    }
    temp = createNodeInDLL();
    temp -> data = x;
    if (position == 1) {
        if (first != NULL) {
            temp -> next = first;
            first -> prev = temp;
        }
        first = temp;
    } else {
        temp -> next = lastNode -> next;
        temp -> prev = lastNode;
        if (lastNode -> next != NULL) {
            lastNode -> next -> prev = temp;
        }
        lastNode -> next = temp;
    }
    return first;
}

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

```
    return first;
}

NODE deleteAtEndInDLL(NODE first) {
    NODE temp, lastNode = first;
    if (lastNode -> next == NULL) {
        first = NULL;
    } else {
        while (lastNode -> next != NULL) {
            temp = lastNode;
            lastNode = lastNode -> next;
        }
        temp -> next = NULL;
    }
    printf("The deleted element from DLL : %d", lastNode -> data);
    free(lastNode);

    return first;
}



NODE deleteAtPositionInDLL(NODE first, int position) {
    NODE prev, lastNode = first;
    if (position == 1) {
        if (lastNode -> next == NULL) {
            first = NULL;
        } else {
            first = first -> next;
            first -> prev = NULL;
        }
    } else {
        int i;
        for (i = 1; i < position; i++) {
            if (lastNode == NULL) {
                printf("No such position in DLL. So deletion is not possible\n");
                return first;
```

```
            }
            prev = lastNode;
            lastNode = lastNode -> next;
        }
        if (lastNode == NULL) {
            printf("No such position in DLL. So deletion is not possible\n");
            return first;
        } else if (lastNode -> next == NULL) {
            prev -> next = NULL;
        } else {
            prev -> next = lastNode -> next;
            prev -> next -> prev = lastNode -> prev;
        }
    }
    printf("The deleted element from DLL : %d\n", lastNode -> data);
    free(lastNode);
    return first;
}
int countInDLL(NODE first) {
    NODE lastNode = first;
    int sum = 0;
    while (lastNode != NULL) {
        sum++;
        lastNode = lastNode -> next;
    }
    return sum;
}
void traverseListInDLL(NODE first) {
    NODE lastNode = first;
    while (lastNode != NULL) {
        printf("%d --> ", lastNode -> data);
        lastNode = lastNode -> next;
    }
    printf("NULL\n");
}

int searchPosOfElementInDLL(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 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\tDOUBLE LINKED LIST IMPLEMENTATION\n\n");
        do{
           printf("\t1.ADD NODES\n\t2.INSERT AT BEGIN\n\t3.INSERT AT END\n\t4.INSERT AT POSITION\n\t5.DELETE AT BEGIN\n\t6.DELETE AT
END\n\t7.DELETE AT POSITION\n\t8.COUNT\n\t9.TRAVERSE LIST\n\t10.SEARCH\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 = addNodesInDLL(first, x);
                    printf("Enter an element : ");
                    scanf("%d", &x);
                 }
                 printf("-------------------------------------------------------------\n");
                 break;
              case 2:

                 printf("Enter an element : ");
                 scanf("%d", &x);
                 first = insertAtBeginInDLL(first, x);



                 printf("-------------------------------------------------------------\n");
                 break;
              case 3:

                 printf("Enter an element : ");
                 scanf("%d", &x);
                 first = insertAtEndInDLL(first, x);
```

```
    printf("------------------------------------------------------------\n");
    break;
case 4:

    printf("Enter a position : ");
    scanf("%d", &pos);
    printf("Enter an element : ");
    scanf("%d", &x);
    if (pos <= 0 || (pos > 1 && first == NULL)) {
        printf("No such position in DLL. So insertion is not possible\n");
    } else {
        first = insertAtPositionInDLL(first, pos, x);
    }


    printf("------------------------------------------------------------\n");
    break;
case 5:

    if (first == NULL) {
        printf("DLL is empty. So "
               "deletion is not possible\n");
    } else {
        first = deleteAtBeginInDLL(first);
    }




    printf("------------------------------------------------------------\n");
    break;
case 6:
    if (first == NULL) {
        printf("DLL is empty. So "
               "deletion is not possible\n");
```

```
    } else {
        first = deleteAtEndInDLL(first);
    }




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

    if (first == NULL) {
        printf("DLL is empty. So deletion is not possible\n");
    } else {
        printf("Enter a position : ");
        scanf("%d", &pos);
        if (pos <= 0) {
            printf("No such position in DLL. So deletion is not possible\n");
        } else {
            first = deleteAtPositionInDLL(first, pos);
        }
    }


    printf("------------------------------------------------------------\n");
    break;
case 8:
    printf("The number of nodes in a DLL are : %d\n", countInDLL(first));



    printf("------------------------------------------------------------\n");
    break;
case 9:
    if (first == NULL) {
        printf("DLL is empty\n");
    } else {
```

```
            printf("The elements in DLL are : ");
            traverseListInDLL(first);
        }



        printf("------------------------------------------------------------\n");
        break;
    case 10:
        printf("Enter a search element : ");
        scanf("%d", &x);
        pos = searchPosOfElementInDLL(first, x);
        if (pos == 0) {
            printf("The given element %d is not found in "
                    "the given DLL.", x);
        } else {
            printf("The given element %d is "
                    "found at position : %d", x, 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);

}
```