Doubly LinkedList

A doubly linked list is an extension of the basic linked list data structure. In a doubly linked list, each node contains a data element and two pointers, one pointing to the next node and another pointing to the previous node.

Algorithms

Let getnode be a function that initializes a new node and returns its pointer.

Creation

1. Start
2. newptr = getnode()
3. if (head == NULL)
   head = newptr
   else
   last->next = nodeptr
   newptr->prev = last
4. last = newptr
5. repeat from 2 if the user wants to add more
6. Stop

Traversing

1. Start
2. ptr = head
3. while (ptr != NULL)
   print ptr->info
   ptr = ptr->next
4. Stop

Insertion

1. Start
2. newptr = getnode()
3. ptr = head
4. while (ptr != NULL AND ptr->info != key)
   ptr = ptr->next
5. if (ptr == NULL)
   print "Key not found"
   else
   newptr->next = ptr->next
   (ptr->next)->prev = newptr
   ptr->next = newptr
   newptr->prev = ptr

6. Stop

Deletion

1. Start
2. ptr = head
3. if(head = NULL)
   print "DLL is empty" and return
4. while (ptr != NULL AND ptr->info != key)
   ptr = ptr->next
5. if (ptr == NULL)
   print "Node with key doesn't exist"
   else
   (ptr-prev)->next = ptr->next
   (ptr->next)->prev = ptr->prev
   if (ptr==head)
   head = head->next
   free(ptr)
6. Stop

IMPLEMENTATION

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

struct Node {
    int info;
    struct Node* next;
    struct Node* prev;
};

struct Node* getnode() {
    struct Node* newptr = (struct Node*)malloc(sizeof(struct Node));
    printf("Enter data for the new node: ");
    scanf("%d", &(newptr->info));
    newptr->next = NULL;
    newptr->prev = NULL;
    return newptr;
}

struct Node* head = NULL;

void createList() {
    char choice;
    struct Node* last = NULL;

    do {
        struct Node* newptr = getnode();
```

```
        if (head == NULL) {
            head = newptr;
        } else {
            last->next = newptr;
            newptr->prev = last;
        }

        last = newptr;

        printf("Do you want to add more nodes? (y/n): ");
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');
}

void traverseList() {
    if (head == NULL) {
        printf("Doubly LinkedList is empty.\n");
        return;
    }

    struct Node* ptr = head;

    while (ptr != NULL) {
        printf("%d ", ptr->info);
        ptr = ptr->next;
    }

    printf("\n");
}

void insertNode(int key) {
    if (head == NULL) {
        printf("Doubly LinkedList is empty. Cannot insert.\n");
        return;
    }

    struct Node* newptr = getnode();
    struct Node* ptr = head;

    while (ptr != NULL && ptr->info != key) {
        ptr = ptr->next;
    }

    if (ptr == NULL) {
        printf("Key not found.\n");
    } else {
        newptr->next = ptr->next;
        if (ptr->next != NULL) {
            (ptr->next)->prev = newptr;
        }
        ptr->next = newptr;
        newptr->prev = ptr;
    }
```

```
}

void deleteNode(int key) {
    if (head == NULL) {
        printf("Doubly LinkedList is empty.\n");
        return;
    }

    struct Node* ptr = head;

    while (ptr != NULL && ptr->info != key) {
        ptr = ptr->next;
    }

    if (ptr == NULL) {
        printf("Node with key %d doesn't exist.\n", key);
    } else {
        if (ptr->prev != NULL) {
            (ptr->prev)->next = ptr->next;
        }
        if (ptr->next != NULL) {
            (ptr->next)->prev = ptr->prev;
        }
        if (ptr == head) {
            head = head->next;
        }
        free(ptr);
    }
}

int main() {
    int ch, x;
    printf("DOUBLY LINKEDLIST IMPLEMENTATION.\n\n");
    createList();
    while(1) {
        printf("1. Insert.\n"
            "2. Remove.\n"
            "3. Print\n"
            "4. Exit\n"
            ">> ");

        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("Enter the key of node after which new node should be
inserted: ");
            scanf("%d", &x);
            insertNode(x);
            break;
        case 2:
            printf("Enter the key of node which is to be removed: ");
            scanf("%d", &x);
```

```
            deleteNode(x);
            break;
        case 3:
            traverseList();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("Invalid option %d\n", ch);
            break;
        }
    }

    return 0;
}
```

OUTPUT

```
DOUBLY LINKEDLIST IMPLEMENTATION.

Enter data for the new node: 10
Do you want to add more nodes? (y/n): y
Enter data for the new node: 20
Do you want to add more nodes? (y/n): n
1. Insert.
2. Remove.
3. Print
4. Exit
>> 1
Enter the key of node after which new node should be inserted: 10
Enter data for the new node: 15
1. Insert.
2. Remove.
3. Print
4. Exit
>> 3
10 15 20
1. Insert.
2. Remove.
3. Print
4. Exit
>> 2
Enter the key of node which is to be removed: 10
1. Insert.
2. Remove.
3. Print
4. Exit
>> 3
```

```
15 20
1. Insert.
2. Remove.
3. Print
4. Exit
>> 1
Enter the key of node after which new node should be inserted: 20
Enter data for the new node: 30
1. Insert.
2. Remove.
3. Print
4. Exit
>> 1
Enter the key of node after which new node should be inserted: 40
Enter data for the new node: 3
Key not found.
1. Insert.
2. Remove.
3. Print
4. Exit
>> 4
```