

LAB-6

Implementation of Doubly Linked List and Circular Linked List Operations in C

Q1) Implement a doubly linked list with basic operations: insert at beginning, insert at end, delete a node, and display the list.

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

struct node {
    int data;
    struct node *prev, *next;
};

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->prev = n->next = NULL;
    return n;
}

void insertBeg(int x) {
    struct node *n = create(x);
    if (!head) head = n;
    else {
        n->next = head;
        head->prev = n;
        head = n;
    }
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) head = n;
    else {
```

```

    struct node *t = head;
    while (t->next) t = t->next;
    t->next = n;
    n->prev = t;
}
}

void deleteNode(int x) {
    struct node *t = head;
    while (t && t->data != x) t = t->next;
    if (!t) return;
    if (t->prev) t->prev->next = t->next;
    else head = t->next;
    if (t->next) t->next->prev = t->prev;
    free(t);
}

void display() {
    struct node *t = head;
    while (t) {
        printf("%d ", t->data);
        t = t->next;
    }
    printf("\n");
}

int main() {
    int ch, x;
    while (1) {
        printf("\n1.Insert at Beginning\n2.Insert at End\n3.Delete
Node\n4.Display\n5.Exit\nEnter choice: ");
        scanf("%d", &ch);
        if (ch == 1) {
            printf("Enter value: ");
            scanf("%d", &x);
            insertBeg(x);
        } else if (ch == 2) {
            printf("Enter value: ");
            scanf("%d", &x);
            insertEnd(x);
        } else if (ch == 3) {
            printf("Enter value to delete: ");

```

```
        scanf("%d", &x);
        deleteNode(x);
    } else if (ch == 4) display();
    else break;
}
return 0;
}
```

Sample Input & Output

1.Insert at Beginning

2.Insert at End

3.Delete Node

4.Display

5.Exit

Enter choice: 1

Enter value: 20

1.Insert at Beginning

2.Insert at End

3.Delete Node

4.Display

5.Exit

Enter choice: 1

Enter value: 34

1.Insert at Beginning

2.Insert at End

3.Delete Node

4.Display

5.Exit

Enter choice: 2034 1

Enter value: 2034

1.Insert at Beginning

2.Insert at End
3.Delete Node
4.Display
5.Exit
Enter choice: 3
Enter value to delete: 2034

1.Insert at Beginning
2.Insert at End
3.Delete Node
4.Display
5.Exit
Enter choice: 4
34 20

1.Insert at Beginning
2.Insert at End
3.Delete Node
4.Display
5.Exit
Enter choice: 5

Q2) Write a function to insert a new node at a specific position in a doubly linked list.

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

struct node {
    int data;
    struct node *prev, *next;
};
```

```

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->prev = n->next = NULL;
    return n;
}

void insertPos(int x, int pos) {
    struct node *n = create(x);
    if (pos == 1) {
        n->next = head;
        if (head) head->prev = n;
        head = n;
        return;
    }
    struct node *t = head;
    for (int i = 1; i < pos - 1 && t; i++)
        t = t->next;
    if (!t) {
        printf("Invalid position\n");
        free(n);
        return;
    }
    n->next = t->next;
    n->prev = t;
    if (t->next) t->next->prev = n;
    t->next = n;
}

void display() {
    struct node *t = head;
    while (t) {
        printf("%d ", t->data);
        t = t->next;
    }
    printf("\n");
}

int main() {
    int n, x, pos;

```

```

printf("Enter number of insertions: ");
scanf("%d", &n);

for (int i = 0; i < n; i++) {
    printf("Enter value and position: ");
    scanf("%d %d", &x, &pos);
    insertPos(x, pos);
    printf("List: ");
    display();
}

return 0;
}

```

Sample Input & Output

```

Enter number of insertions: 3
Enter value and position: 34
1
List: 34
Enter value and position: 20
2
List: 34 20
Enter value and position: 2034
0
List: 34 2034 20

```

Q3) Write a function to delete the first occurrence of a given value from a doubly linked list.

```

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

struct node {
    int data;
    struct node *prev, *next;
};

```

```

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->prev = n->next = NULL;
    return n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) head = n;
    else {
        struct node *t = head;
        while (t->next) t = t->next;
        t->next = n;
        n->prev = t;
    }
}

void deleteVal(int x) {
    struct node *t = head;
    while (t && t->data != x)
        t = t->next;
    if (!t) {
        printf("Value not found\n");
        return;
    }
    if (t->prev)
        t->prev->next = t->next;
    else
        head = t->next;
    if (t->next)
        t->next->prev = t->prev;
    free(t);
}

void display() {
    struct node *t = head;
    while (t) {
        printf("%d ", t->data);
    }
}

```

```
        t = t->next;
    }
    printf("\n");
}

int main() {
    int n, x, del;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("Current list: ");
    display();

    printf("Enter value to delete: ");
    scanf("%d", &del);
    deleteVal(del);

    printf("After deletion: ");
    display();

    return 0;
}
```

Sample Input & Output

Enter number of elements: 4

Enter value: 34

Enter value: 20

Enter value: 2034

Enter value: 3420

Current list: 34 20 2034 3420

Enter value to delete: 2034

After deletion: 34 20 3420

Q4) Implement a function to reverse a doubly linked list in place.

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

struct node {
    int data;
    struct node *prev, *next;
};

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->prev = n->next = NULL;
    return n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) head = n;
    else {
        struct node *t = head;
        while (t->next) t = t->next;
        t->next = n;
        n->prev = t;
    }
}

void reverse() {
    struct node *t = head, *temp = NULL;
    if (!head) return;
    while (t) {
        temp = t->prev;
        t->prev = t->next;
        t->next = temp;
        t = t->prev;
    }
    if (temp) head = temp->prev;
}
```

```

}

void display() {
    struct node *t = head;
    while (t) {
        printf("%d ", t->data);
        t = t->next;
    }
    printf("\n");
}

int main() {
    int n, x;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("Original list: ");
    display();

    reverse();

    printf("Reversed list: ");
    display();

    return 0;
}

```

Sample Input & Output

Enter number of elements: 4

Enter value: 20

Enter value: 2034

Enter value: 34

Enter value: 3420

Original list: 20 2034 34 3420

Reversed list: 3420 34 2034 20

Q5) Write a function to find the length and the middle node of a doubly linked list.

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

struct node {
    int data;
    struct node *prev, *next;
};

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->prev = n->next = NULL;
    return n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) head = n;
    else {
        struct node *t = head;
        while (t->next) t = t->next;
        t->next = n;
        n->prev = t;
    }
}

int length() {
    int c = 0;
    struct node *t = head;
    while (t) {
        c++;
        t = t->next;
    }
}
```

```

    return c;
}

struct node* middle() {
    struct node *slow = head, *fast = head;
    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
    }
    return slow;
}

void display() {
    struct node *t = head;
    while (t) {
        printf("%d ", t->data);
        t = t->next;
    }
    printf("\n");
}

int main() {
    int n, x;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("List: ");
    display();

    int len = length();
    printf("Length = %d\n", len);

    struct node *mid = middle();
    if (mid) printf("Middle node = %d\n", mid->data);

    return 0;
}

```

}

Sample Input & Output

Enter number of elements: 4

Enter value: 3420

Enter value: 30 2034

Enter value: 34

Enter value: 20

List: 3420 2034 34 20

Length = 4

Middle node = 34

Q6) Implement a circular singly linked list with insertion at the beginning, at the end, and display operations.

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

```
struct node {
    int data;
    struct node *next;
};
```

```
struct node *head = NULL;
```

```
struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->next = NULL;
    return n;
}
```

```
void insertBeg(int x) {
    struct node *n = create(x);
    if (!head) {
        head = n;
```

```

        n->next = head;
        return;
    }
    struct node *t = head;
    while (t->next != head)
        t = t->next;
    n->next = head;
    t->next = n;
    head = n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) {
        head = n;
        n->next = head;
        return;
    }
    struct node *t = head;
    while (t->next != head)
        t = t->next;
    t->next = n;
    n->next = head;
}

void display() {
    if (!head) {
        printf("List empty\n");
        return;
    }
    struct node *t = head;
    do {
        printf("%d ", t->data);
        t = t->next;
    } while (t != head);
    printf("\n");
}

int main() {
    int ch, x;
    while (1) {
        printf("\n1.Insert at Beginning\n2.Insert at End\n3.Display\n4.Exit\nEnter

```

```
choice: ");
scanf("%d", &ch);
if (ch == 1) {
    printf("Enter value: ");
    scanf("%d", &x);
    insertBeg(x);
} else if (ch == 2) {
    printf("Enter value: ");
    scanf("%d", &x);
    insertEnd(x);
} else if (ch == 3) display();
else break;
}
return 0;
}
```

Sample Input & Output

1.Insert at Beginning

2.Insert at End

3.Display

4.Exit

Enter choice: 1

Enter value: 34

1.Insert at Beginning

2.Insert at End

3.Display

4.Exit

Enter choice: 1

Enter value: 20

1.Insert at Beginning

2.Insert at End

3.Display

4.Exit

Enter choice: 2

Enter value: 2034

1.Insert at Beginning

2.Insert at End

3.Display

4.Exit

Enter choice: 1

Enter value: 3420

1.Insert at Beginning

2.Insert at End

3.Display

4.Exit

Enter choice: 3

3420 20 34 2034

1.Insert at Beginning

2.Insert at End

3.Display

4.Exit

Enter choice: 4

Q7) Write a function to insert a new node after a given node in a circular linked list.

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

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

```
struct node {
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
struct node *head = NULL;
```

```

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->next = NULL;
    return n;
}

```

```

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) {
        head = n;
        n->next = head;
        return;
    }
    struct node *t = head;
    while (t->next != head)
        t = t->next;
    t->next = n;
    n->next = head;
}

```

```

void insertAfter(int key, int x) {
    if (!head) return;
    struct node *t = head;
    do {
        if (t->data == key) {
            struct node *n = create(x);
            n->next = t->next;
            t->next = n;
            return;
        }
        t = t->next;
    } while (t != head);
    printf("Key not found\n");
}

```

```

void display() {
    if (!head) {
        printf("List empty\n");
        return;
    }
}

```

```

struct node *t = head;
do {
    printf("%d ", t->data);
    t = t->next;
} while (t != head);
printf("\n");
}

int main() {
    int n, x, key;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("Current list: ");
    display();

    printf("Enter key after which to insert: ");
    scanf("%d", &key);
    printf("Enter value to insert: ");
    scanf("%d", &x);
    insertAfter(key, x);

    printf("After insertion: ");
    display();

    return 0;
}

```

Sample Input & Output

Enter number of elements: 4

Enter value: 34

Enter value: 20

Enter value: 3420

Enter value: 2034

Current list: 34 20 3420 2034

Enter key after which to insert: 20

Enter value to insert: 60 80

After insertion: 34 20 680 3420 2034

Q8) Write a function to delete a node with a specific value from a circular linked list.

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

struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->next = NULL;
    return n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) {
        head = n;
        n->next = head;
        return;
    }
    struct node *t = head;
    while (t->next != head)
        t = t->next;
    t->next = n;
    n->next = head;
}
```

```

void deleteVal(int x) {
    if (!head) {
        printf("List empty\n");
        return;
    }
    struct node *curr = head, *prev = NULL;

    // If only one node
    if (head->next == head && head->data == x) {
        free(head);
        head = NULL;
        return;
    }

    // If head node is to be deleted
    if (head->data == x) {
        struct node *t = head;
        while (t->next != head)
            t = t->next;
        t->next = head->next;
        struct node *temp = head;
        head = head->next;
        free(temp);
        return;
    }

    // Search and delete
    do {
        if (curr->data == x) {
            prev->next = curr->next;
            free(curr);
            return;
        }
        prev = curr;
        curr = curr->next;
    } while (curr != head);

    printf("Value not found\n");
}

void display() {
    if (!head) {

```

```

        printf("List empty\n");
        return;
    }
    struct node *t = head;
    do {
        printf("%d ", t->data);
        t = t->next;
    } while (t != head);
    printf("\n");
}

int main() {
    int n, x, del;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("Current list: ");
    display();

    printf("Enter value to delete: ");
    scanf("%d", &del);
    deleteVal(del);

    printf("After deletion: ");
    display();

    return 0;
}

```

Sample Input & Output

Enter number of elements: 4

Enter value: 20

Enter value: 34

Enter value: 3420

Enter value: 4 3 2034

Current list: 20 34 3420 2034

Enter value to delete: 2034

After deletion: 20 34 3420

Q9) Write a function to check whether a given linked list is circular or not.

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

struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
    n->next = NULL;
    return n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) {
        head = n;
        return;
    }
    struct node *t = head;
    while (t->next)
        t = t->next;
    t->next = n;
}

int isCircular() {
    if (!head) return 0;
    struct node *t = head->next;
```

```

while (t && t != head)
    t = t->next;
return (t == head);
}

void makeCircular() {
    if (!head) return;
    struct node *t = head;
    while (t->next)
        t = t->next;
    t->next = head;
}

void display(int limit) {
    struct node *t = head;
    int count = 0;
    while (t && count < limit) {
        printf("%d ", t->data);
        t = t->next;
        count++;
    }
    printf("\n");
}

int main() {
    int n, x, choice;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("List created.\n1.Check if circular\n2.Make circular and check\nagain\nEnter choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        if (isCircular()) printf("List is circular\n");
        else printf("List is not circular\n");
    }
}

```

```
} else if (choice == 2) {
    makeCircular();
    if (isCircular()) printf("List is circular\n");
    else printf("List is not circular\n");
}

return 0;
}
```

Sample Input & Output

```
Enter number of elements: 4
Enter value: 20
Enter value: 34
Enter value: 3420
Enter value: 680
List created.
1.Check if circular
2.Make circular and check again
Enter choice: 1
List is not circular
```

Q10) Write a program to split a circular linked list into two halves and display both lists.

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

struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;

struct node* create(int x) {
    struct node *n = malloc(sizeof(struct node));
    n->data = x;
```

```

    n->next = NULL;
    return n;
}

void insertEnd(int x) {
    struct node *n = create(x);
    if (!head) {
        head = n;
        n->next = head;
        return;
    }
    struct node *t = head;
    while (t->next != head)
        t = t->next;
    t->next = n;
    n->next = head;
}

void display(struct node *h) {
    if (!h) {
        printf("List empty\n");
        return;
    }
    struct node *t = h;
    do {
        printf("%d ", t->data);
        t = t->next;
    } while (t != h);
    printf("\n");
}

void splitList(struct node *head, struct node **head1, struct node **head2) {
    if (!head || head->next == head) {
        *head1 = head;
        *head2 = NULL;
        return;
    }

    struct node *slow = head, *fast = head;

    while (fast->next != head && fast->next->next != head) {
        fast = fast->next->next;
    }

```

```

    slow = slow->next;
}

*head1 = head;
*head2 = slow->next;
fast = (fast->next->next == head) ? fast->next : fast;
fast->next = *head2;
slow->next = *head1;
}

int main() {
    int n, x;
    struct node *head1 = NULL, *head2 = NULL;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter value: ");
        scanf("%d", &x);
        insertEnd(x);
    }

    printf("Original circular list: ");
    display(head);

    splitList(head, &head1, &head2);

    printf("First half: ");
    display(head1);

    printf("Second half: ");
    display(head2);

    return 0;
}

```

Sample Input & Output

Enter number of elements: 4

Enter value: 34

Enter value: 3420

Enter value: 2034

Enter value: 20

Original circular list: 34 3420 2034 20

First half: 34 3420

Second half: 2034 20