5)Doubly Linked list implementation

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

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

struct node *head = NULL, *last = NULL, *newn, *trav;

void create_list() {
    int value;
    newn = (struct node *)malloc(sizeof(struct node));
    printf("\nEnter value: ");
    scanf("%d", &value);

    newn->data = value;
    newn->next = NULL;

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

```
    }
}

void insert_at_begning(int value) {
    newn = (struct node *)malloc(sizeof(struct node));
    newn->data = value;

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

void insert_at_end(int value) {
    newn = (struct node *)malloc(sizeof(struct node));
    newn->data = value;

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

```
        newn->next = NULL;
        newn->prev = last;
        last->next = newn;
        last = newn;
    }
}

int insert_at_middle() {
    if (last == NULL || last->prev == NULL) {
        printf("\nAt least two elements are needed for middle insertion.\n");
        return 0;
    }

    int loc, value;
    printf("\nEnter location for insertion: ");
    scanf("%d", &loc);
    printf("\nEnter value to be inserted: ");
    scanf("%d", &value);

    newn = (struct node *)malloc(sizeof(struct node));
    newn->data = value;

    struct node *temp = head;
    while (temp->data != loc) {
        temp = temp->next;
        if (temp == NULL) {
            printf("\nSorry, element %d not found.\n", loc);
            return 0;
```

```
        }
    }

    struct node *var2 = temp->next;
    temp->next = newn;
    newn->next = var2;
    newn->prev = temp;

    if (var2 != NULL) {
        var2->prev = newn;
    }

    return 0;
}

void delete_from_front() {
    if (head == NULL) {
        printf("No elements for deletion in the list.\n");
    } else if (head->next == NULL) {
        printf("Deleted element: %d\n", head->data);
        head = last = NULL;
    } else {
        struct node *temp = head->next;
        printf("Deleted element: %d\n", head->data);
        head->next = NULL;
        temp->prev = NULL;
        head = temp;
    }
```

```
}

void delete_from_end() {
    if (last == NULL) {
        printf("No elements for deletion in the list.\n");
    } else if (last->prev == NULL) {
        printf("Deleted element: %d\n", last->data);
        head = last = NULL;
    } else {
        printf("Deleted element: %d\n", last->data);
        last = last->prev;
        last->next = NULL;
    }
}

void delete_from_middle() {
    if (last == NULL || last->prev == NULL) {
        printf("At least two elements are needed for middle deletion.\n");
        return;
    }

    int value;
    printf("\nEnter the data to be deleted: ");
    scanf("%d", &value);

    struct node *temp = head;

    while (temp != NULL) {
```

```
    if (temp->data == value) {
      if (temp == head) {
        printf("Cannot delete the first node using middle deletion. Enter middle nodes only.\n");
        return;
      }

      struct node *var = temp->prev;
      var->next = temp->next;

      if (temp->next != NULL) {
        temp->next->prev = var;
      } else {
        last = var;
      }

      free(temp);
      printf("Deleted element: %d\n", value);
      return;
    } else {
      temp = temp->next;
    }
  }

  printf("%d is not available. Enter only middle elements.\n", value);
}

void display() {
```

```
    trav = last; // head;
    if (trav == NULL) {
        printf("\nList is Empty\n");
        return;
    } else {
        printf("\n\nElements in the List: ");
        while (trav != NULL) {
            printf("%d<--> ", trav->data);
            trav = trav->prev;  // next;
        }
        printf("\n");
    }
}

int main() {
    int ch;
    char ch1;

    while (1) {
        printf("\nDouble Linked List Operations");
        printf("\n1. Create Double Linked List");
        printf("\n2. Insertion at the beginning");
        printf("\n3. Insertion at the end");
        printf("\n4. Insertion at the middle");
        printf("\n5. Deletion from the front");
        printf("\n6. Deletion from the end");
        printf("\n7. Deletion of the middle data");
        printf("\n8. Display");
```

```
printf("\n9. Exit\n");
printf("\nEnter the choice: ");
scanf("%d", &ch);

switch (ch) {
   case 1:
      do {
         create_list();
         display();
         printf("Do you want to create list? (y/n): ");
         getchar();
         scanf("%c", &ch1);
      } while (ch1 == 'y');
      break;
   case 2:
      {
         int value;
         printf("\nEnter the value to be inserted: ");
         scanf("%d", &value);
         insert_at_begning(value);
         display();
      }
      break;
   case 3:
      {
         int value;
         printf("\nEnter value to be inserted: ");
         scanf("%d", &value);
```

```
            insert_at_end(value);
            display();
        }
        break;
    case 4:
        insert_at_middle();
        display();
        break;
    case 5:
        delete_from_front();
        display();
        break;
    case 6:
        delete_from_end();
        display();
        break;
    case 7:
        display();
        delete_from_middle();
        display();
        break;
    case 8:
        display();
        break;
    case 9:
    printf("EXIT POINT\n");
        exit(0);
}
```

```
    }

    return 0;
}
```

Output

```
Double Linked List Operations
1. Create Double Linked List
2. Insertion at the beginning
3. Insertion at the end
4. Insertion at the middle
5. Deletion from the front
6. Deletion from the end
7. Deletion of the middle data
8. Display
9. Exit

Enter the choice: 1
Enter value: 3
Elements in the List: 3<-->
Do you want to create list? (y/n): n
Double Linked List Operations
1. Create Double Linked List
2. Insertion at the beginning
3. Insertion at the end
4. Insertion at the middle
5. Deletion from the front
6. Deletion from the end
7. Deletion of the middle data
```

8. Display

9. Exit

Enter the choice: 2

Enter the value to be inserted: 1

Elements in the List: 3<--> 1<-->

Double Linked List Operations

1. Create Double Linked List

2. Insertion at the beginning

3. Insertion at the end

4. Insertion at the middle

5. Deletion from the front

6. Deletion from the end

7. Deletion of the middle data

8. Display

9. Exit

Enter the choice: 3

Enter value to be inserted: 6

Elements in the List: 6<--> 3<--> 1<-->

Double Linked List Operations

1. Create Double Linked List

2. Insertion at the beginning

3. Insertion at the end

4. Insertion at the middle

5. Deletion from the front

6. Deletion from the end

7. Deletion of the middle data

8. Display

9. Exit

Enter the choice: 7

Elements in the List: 6<--> 3<--> 1<-->

Enter the data to be deleted: 3

Deleted element: 3

Elements in the List: 6<--> 1<-->

Double Linked List Operations

1. Create Double Linked List

2. Insertion at the beginning

3. Insertion at the end

4. Insertion at the middle

5. Deletion from the front

6. Deletion from the end

7. Deletion of the middle data

8. Display

9. Exit

Enter the choice: 2

Enter the value to be inserted: 5

Elements in the List: 6<--> 1<--> 5<-->

Double Linked List Operations

1. Create Double Linked List
2. Insertion at the beginning
3. Insertion at the end
4. Insertion at the middle
5. Deletion from the front
6. Deletion from the end
7. Deletion of the middle data
8. Display
9. Exit

Enter the choice: 4

Enter location for insertion: 1

Enter value to be inserted: 7

Elements in the List: 6<--> 7<--> 1<--> 5<-->

Double Linked List Operations

1. Create Double Linked List
2. Insertion at the beginning
3. Insertion at the end
4. Insertion at the middle
5. Deletion from the front
6. Deletion from the end
7. Deletion of the middle data
8. Display
9. Exit

Enter the choice: 5

Deleted element: 5

Elements in the List: 6<--> 7<--> 1<-->

Double Linked List Operations

1. Create Double Linked List
2. Insertion at the beginning
3. Insertion at the end
4. Insertion at the middle
5. Deletion from the front
6. Deletion from the end
7. Deletion of the middle data
8. Display
9. Exit

Enter the choice: 6

Deleted element: 6

Elements in the List: 7<--> 1<-->

Double Linked List Operations

1. Create Double Linked List
2. Insertion at the beginning
3. Insertion at the end
4. Insertion at the middle

5. Deletion from the front

6. Deletion from the end

7. Deletion of the middle data

8. Display

9. Exit

Enter the choice: 8

Elements in the List: 7<--> 1<-->

Double Linked List Operations

1. Create Double Linked List

2. Insertion at the beginning

3. Insertion at the end

4. Insertion at the middle

5. Deletion from the front

6. Deletion from the end

7. Deletion of the middle data

8. Display

9. Exit

Enter the choice: 9

EXIT POINT