EXPERIMENT NO. 8

```
//The Code Is As Follows :-


#include <stdio.h>

#include <stdlib.h>

                                                        /*Define structure for a
                                                        doubly linked list node*/

struct Node {

   int data;                                             // Data stored in the node

   struct Node* next;                                    // Pointer to the next node

   struct Node* prev;                              // Pointer to the previous node

};


                                           /*Function to insert a new node at the
                                           beginning of the doubly linked list*/

void insertAtBeginning(struct Node** head_ref, int new_data) {

                                           // Allocate memory for the new node

   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));


                                                     // Assign data to the new node

   new_node->data = new_data;


                                                        /* Make next of new node
                                                   as head and previous as NULL*/

   new_node->next = (*head_ref);
```

```
    new_node->prev = NULL;

                                                    /*Change prev of head
                                                    node to new node*/
    if ((*head_ref) != NULL)  // If list is not empty
        (*head_ref)->prev = new_node;

                                                    /*Move the head to point to
                                                    the new node*/
    (*head_ref) = new_node;
}

                                                    /*Function to delete a node
                                                    from the doubly linked list*/
void deleteNode(struct Node** head_ref, int key) {
                                                    // If the list is empty
    if (*head_ref == NULL) return;

                                                    /*Temporary pointers for
                                                          traversal*/
    struct Node *temp = *head_ref, *prev = NULL;

                                                    /*If the node to be deleted
                                                    is the head node*/
    if (temp != NULL && temp->data == key) {
        *head_ref = temp->next;                     // Change head
        if (*head_ref != NULL)
```

```
            (*head_ref)->prev = NULL;
        free(temp);                                  // Free old head
        return;
    }
                                                     /* Search for the key to be
                                                             deleted*/
    while (temp != NULL && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }


                                                             /* If key was not
                                                     present in linked list*/
    if (temp == NULL) return;
                                                             /*Unlink the node
                                                             from linked list*/
    if (temp->next != NULL)
        temp->next->prev = temp->prev;


    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    free(temp);                                              // Free memory
}


                                                             /*This function prints
                                                             contents of linked
                                              list starting from the given node*/
```

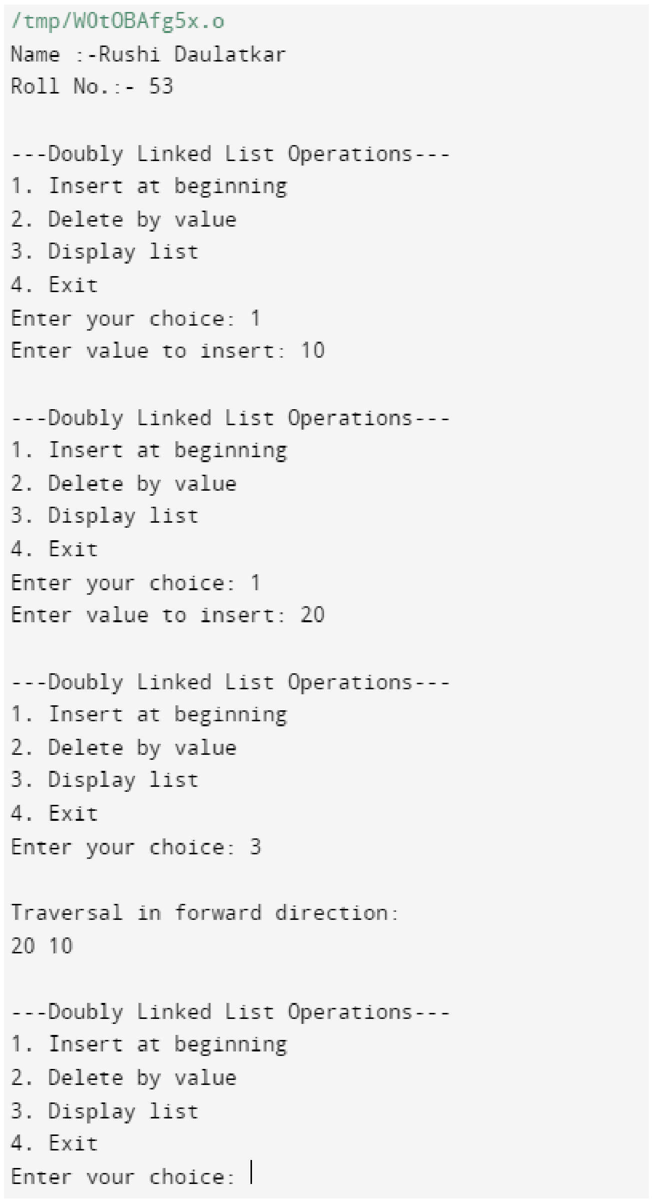
```
void displayList(struct Node* node) {
    printf("\nTraversal in forward direction: \n");
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
    printf("\n");
}

int main() {
    struct Node* head = NULL;
    int choice, data;

                                                  /*Menu-driven loop for
                                          performing operations on the linked list*/
    while(1) {
        printf("\n---Doubly Linked List Operations---\n");
        printf("1. Insert at beginning\n");
        printf("2. Delete by value\n");
        printf("3. Display list\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

                                          // Switch case to handle user choice
        switch(choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &data);
                insertAtBeginning(&head, data);
                break;
            case 2:
                printf("Enter value to delete: ");
                scanf("%d", &data);
                deleteNode(&head, data);
                break;
            case 3:
                displayList(head);
                break;
            case 4:
                printf("Exiting program.\n");
                return 0;
            default:
                printf("Invalid choice, please try again.\n");
        }
    }
    return 0;
}                                                                       //END OF THE PROGRAM
```

OUTPUT :-



```
/tmp/WOtOBAfg5x.o
Name :-Rushi Daulatkar
Roll No.:- 53

---Doubly Linked List Operations---
1. Insert at beginning
2. Delete by value
3. Display list
4. Exit
Enter your choice: 1
Enter value to insert: 10

---Doubly Linked List Operations---
1. Insert at beginning
2. Delete by value
3. Display list
4. Exit
Enter your choice: 1
Enter value to insert: 20

---Doubly Linked List Operations---
1. Insert at beginning
2. Delete by value
3. Display list
4. Exit
Enter your choice: 3

Traversal in forward direction:
20 10

---Doubly Linked List Operations---
1. Insert at beginning
2. Delete by value
3. Display list
4. Exit
Enter your choice: 
```