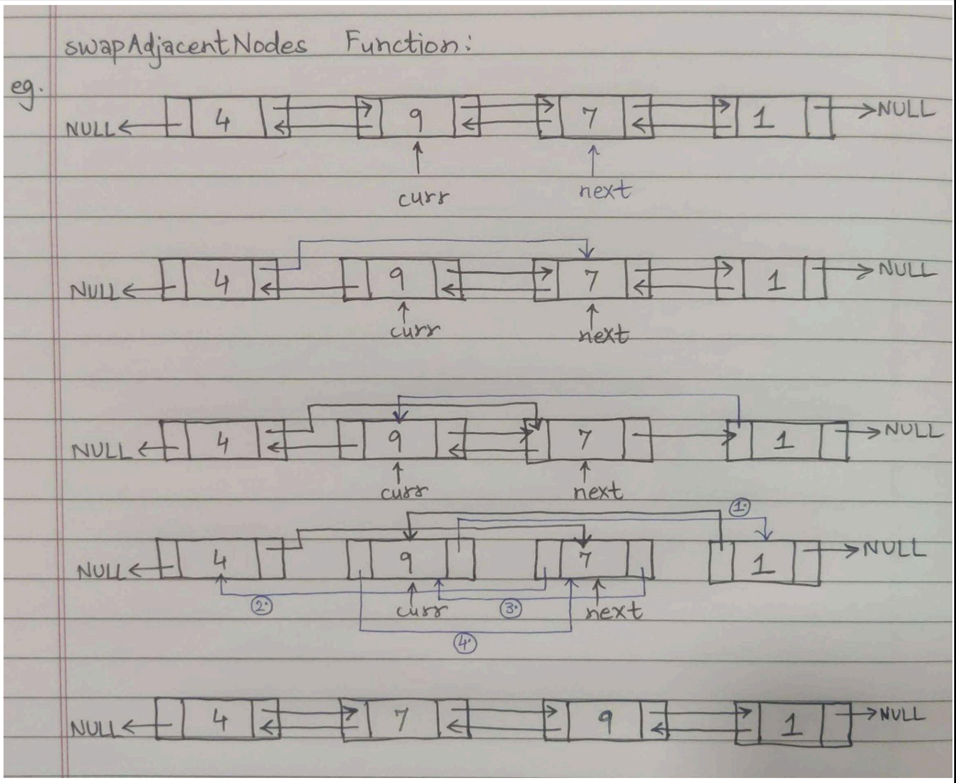
Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of AI-ML

Experiment	4
Aim	Implement the given problem statement of a doubly linked list.
Objective	Given a doubly linked list , sort it (arrange the values in ascending order). Avoid the use of any data structures other than doubly linked lists. Sample Input: 6 < - > 5 < - > 3 < - > 8 < - > 7 < - > 1 Output: 1 < - > 3 < - > 5 < - > 6 < - > 7 < - > 8
Name	Balla Mahadev Shrikrishna
UCID	2023300010
Class	A
Batch	A
Date of Submission	11-09-24

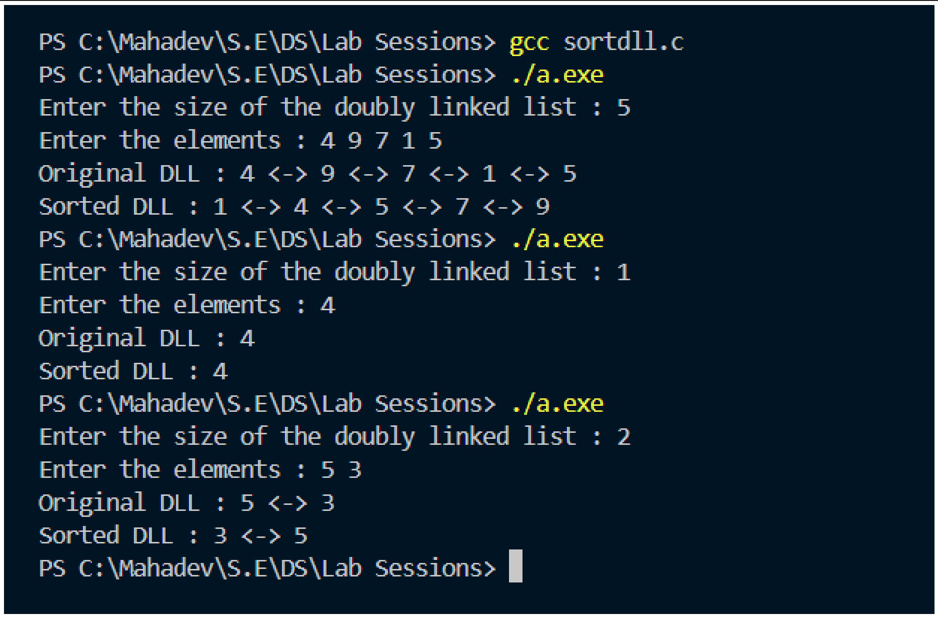
Explanation of the technique used	 swapAdjacentNodes Function: eg. NULL 4 9 7 1 NULL curr next 1. 2. 3. 4.

Program(Code)	#include<stdio.h>
	#include<stdlib.h>
	
	struct Node{
	    struct Node *right;
	    struct Node *left;

```
    int val;
};

struct Node *createNode(int val){
    struct Node *new = (struct Node*)malloc(sizeof(struct Node));
    new->right = NULL;
    new->left = NULL;
    new->val = val;
    return new;
}

struct Node *insertAtEnd(struct Node *head, int val){
    struct Node *newNode = createNode(val);
    if(head == NULL){
        return newNode;
    }
    else{
        struct Node *temp = head;
        while (temp->right != NULL){
            temp = temp->right;
        }
        temp->right = newNode;
        newNode->left = temp;
        return head;
    }
}

void freeLL(struct Node* head){
    struct Node *temp = NULL;
    while(head != NULL){
        temp = head;
        head = head->right;
        free(temp);
    }
}

void swapAdjacentNodes(struct Node **head, struct Node *curr){
    struct Node *next = curr->right;
    if(curr->left != NULL){
        curr->left->right = next;
    }
    else{
        *head = next;
    }
    if(next->right != NULL){
        next->right->left = curr;
    }

    curr->right = next->right;
    next->left = curr->left;
```

```c
    next->right = curr;
    curr->left = next;
}

struct Node *sortdll(struct Node *head) {
    if (head == NULL || head->right == NULL) {
        return head; // Empty or single-node list
    }

    int swapped;
    struct Node *temp;
    do{
        swapped = 0;
        temp = head;

        while(temp->right != NULL){
            if(temp->val > temp->right->val){
                swapAdjacentNodes(&head, temp);
                swapped = 1;
            }
            else{
                temp = temp->right;
            }
        }
    }
    while(swapped);
    return head;
}

void printList(struct Node *head){
    while(head->right != NULL){
        printf("%d->", head->val);
        head = head->right;
    }
    printf("%d", head->val);
}

int main(){
    struct Node *head = NULL; int size, val;
    printf("Enter the size of the doubly linked list : ");
    scanf("%d", &size);
    printf("Enter the elements : ");
    for(int i=0; i<size; i++){
        scanf("%d",&val);
        head = insertAtEnd(head, val);
    }
    printf("Original DLL : ");
    printList(head);
    printf("\n");
    head = sortdll(head);
    printf("Sorted DLL : ");
    printList(head);
    printf("\n");

    freeLL(head);
    return 0;
}
```

Output



```
PS C:\Mahadev\S.E\DS\Lab Sessions> gcc sortdll.c
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter the size of the doubly linked list : 5
Enter the elements : 4 9 7 1 5
Original DLL : 4 <-> 9 <-> 7 <-> 1 <-> 5
Sorted DLL : 1 <-> 4 <-> 5 <-> 7 <-> 9
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter the size of the doubly linked list : 1
Enter the elements : 4
Original DLL : 4
Sorted DLL : 4
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter the size of the doubly linked list : 2
Enter the elements : 5 3
Original DLL : 5 <-> 3
Sorted DLL : 3 <-> 5
PS C:\Mahadev\S.E\DS\Lab Sessions> 
```

Conclusion

In this experiment, I implemented a sorting algorithm for a doubly linked list. This exercise enhanced my understanding of manipulating linked lists and applying sorting algorithms.