Name:

Wajiha Zahid

Roll No:

S24-040

Subject:

“DSA LAB ”

Section

BSSE-3A

Resource Person:

Sir Rasikh Ali

(Lab Task 8)

Q1: Merge two LinkedLists

Task:

1. Create 2 Singly LinkedLists and Merge them and display them.

2. Create 2 Double LinkedLists and Merge them and display them.

Answer:

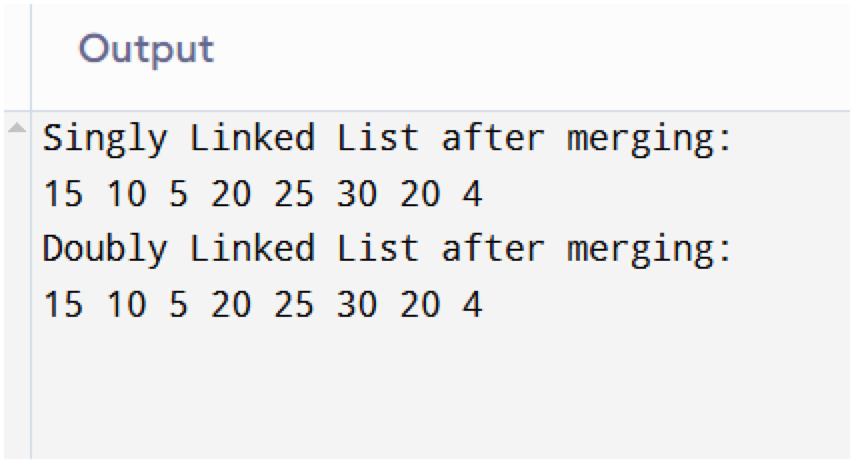
```
#include<iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
struct DNode {
    int data;
    DNode* prev;
    DNode* next;
};

void insertAtBeginning(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}
void insertAtEnd(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    Node* last = *head_ref;
    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL) {
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
}
void insertAtBeginningD(DNode** head_ref, int new_data) {
    DNode* new_node = new DNode();
    new_node->data = new_data;
    new_node->prev = NULL;
    new_node->next = (*head_ref);

    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;

    (*head_ref) = new_node;
}
void insertAtEndD(DNode** head_ref, int new_data) {
    DNode* new_node = new DNode();
    DNode* last = *head_ref;
    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL) {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
    new_node->prev = last;
}
```

```
Node* mergeLists(Node* a, Node* b) {
    Node* result = NULL;

    if (a == NULL)
        return(b);
    else if (b == NULL)
        return(a);

    if (a->data <= b->data) {
        result = a;
        result->next = mergeLists(a->next, b);
    } else {
        result = b;
        result->next = mergeLists(a, b->next);
    }
    return(result);
}
DNode* mergeListsD(DNode* a, DNode* b) {
    DNode* result = NULL;

    if (a == NULL)
        return(b);
    else if (b == NULL)
        return(a);
    if (a->data <= b->data) {
        result = a;
        result->next = mergeListsD(a->next, b);
        if (result->next != NULL)
            result->next->prev = result;
    } else {
        result = b;
        result->next = mergeListsD(a, b->next);
        if (result->next != NULL)
            result->next->prev = result;
    }
    return(result);
}
void printList(Node* node) {
    while (node != NULL) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}
void printListD(DNode* node) {
    while (node != NULL) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
    while (node != NULL) {
        cout << node->data << " ";
        node = node->prev;
    }
    cout << endl;
}

int main() {
    Node* head1 = NULL;
    Node* head2 = NULL;
    DNode* headD1 = NULL;
    DNode* headD2 = NULL;

    // Insert nodes in Singly Linked List 1
    insertAtBeginning(&head1, 15);
    insertAtEnd(&head1, 10);
    insertAtEnd(&head1, 5);
    insertAtEnd(&head1, 20);

    // Insert nodes in Singly Linked List 2
    insertAtBeginning(&head2, 25);
    insertAtEnd(&head2, 30);
    insertAtEnd(&head2, 20);
    insertAtEnd(&head2, 4);
```

```
// Insert nodes in Doubly Linked List 1
insertAtBeginningD(&headD1, 15);
insertAtEndD(&headD1, 10);
insertAtEndD(&headD1, 5);
insertAtEndD(&headD1, 20);

// Insert nodes in Doubly Linked List 2
insertAtBeginningD(&headD2, 25);
insertAtEndD(&headD2, 30);
insertAtEndD(&headD2, 20);
insertAtEndD(&headD2, 4);

// Merge Singly Linked List 1 and 2
Node* mergedList = mergeLists(head1, head2);

// Merge Doubly Linked List 1 and 2
DNode* mergedListD = mergeListsD(headD1, headD2);

// Print merged Singly Linked List
cout << "Singly Linked List after merging:" << endl;
printList(mergedList);

// Print merged Doubly Linked List
cout << "Doubly Linked List after merging:" << endl;
```

```
    printListD(mergedListD);


    return 0;
}
```

Output:



```
Singly Linked List after merging:
15 10 5 20 25 30 20 4
Doubly Linked List after merging:
15 10 5 20 25 30 20 4
```