**Week 5:** **Doubly Linked List & Sorting**

1. **Aim:** Implement Insert and Delete functions of a doubly linked list.

**Input Format**

* Input lines contain choices for insertion at front and end, deletion from front and end. For example, 1. insert at front 2. insert at end 3, delete front, 4. delete last, 5.display and 0 for exit.
* For insertion functions, value must be given along with choice 1 and 2.
* 0 must be given at end of input.

**Program:**

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

class Node

{

public:

    int val;

    Node \*next;

    Node \*prev;

    Node()

    {

        val = 0;

        next = NULL;

        prev = NULL;

    }

    Node(int x)

    {

        val = x;

        next = NULL;

        prev = NULL;

    }

};

void insert\_at\_front(Node \*&head, Node \*&tail, int x)

{

    Node \*temp = new Node(x);

    if (head == NULL)

    {

        head = temp;

        tail = temp;

        return;

    }

    temp->next = head;

    head->prev = temp;

    head = temp;

}

void insert\_at\_end(Node \*&head, Node \*&tail, int x)

{

    Node \*temp = new Node(x);

    if (head == NULL)

    {

        head = temp;

        tail = temp;

        return;

    }

    tail->next = temp;

    temp->prev = tail;

    tail = temp;

}

void delete\_first(Node \*&head, Node \*&tail)

{

    if (head == NULL)

    {

        cout << "No node in the list to delete" << endl;

        return;

    }

    else if (head->next == NULL)

    {

        head = NULL;

        tail = NULL;

        return;

    }

    head = head->next;

    head->prev = NULL;

}

void delete\_l(Node \*&head, Node \*&tail)

{

    if (head == NULL)

        return;

    else if (head == tail)

    {

        head == NULL;

        tail == NULL;

        return;

    }

    tail = tail->prev;

    tail->next = NULL;

}

void display(Node \*head)

{

    Node \*temp = head;

    while (temp != NULL)

    {

        cout << temp->val << " ";

        temp = temp->next;

    }

    cout << endl;

}

int main()

{

    /\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

    Node \*head = NULL;

    Node \*tail = NULL;

    int t;

    cin >> t;

    while (t != 0)

    {

        if (t == 1)

        {

            int n;

            cin >> n;

            insert\_at\_front(head, tail, n);

        }

        else if (t == 2)

        {

            int n;

            cin >> n;

            insert\_at\_end(head, tail, n);

        }

        else if (t == 3)

        {

            delete\_first(head, tail);

        }

        else if (t == 4)

        {

            delete\_l(head, tail);

        }

        else if (t == 5)

        {

            display(head);

        }

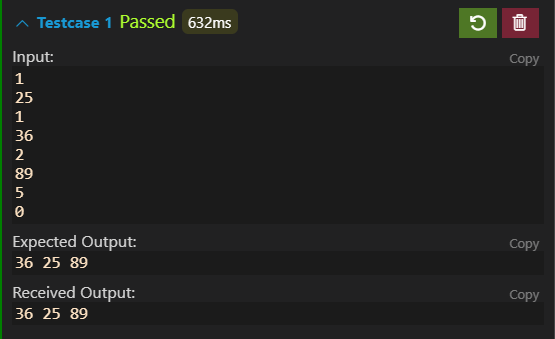
        cin >> t;

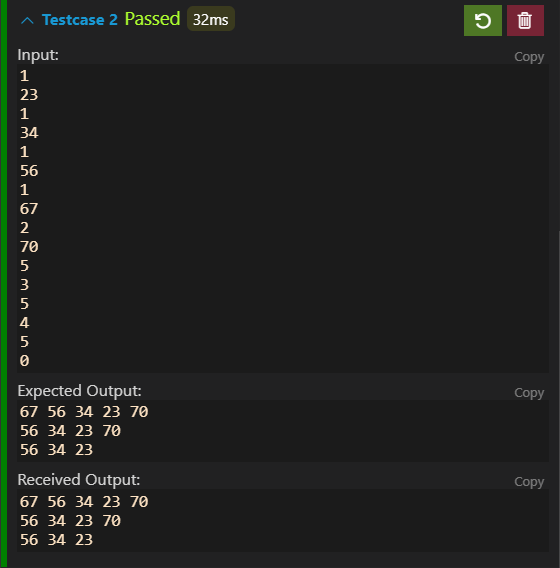
    }

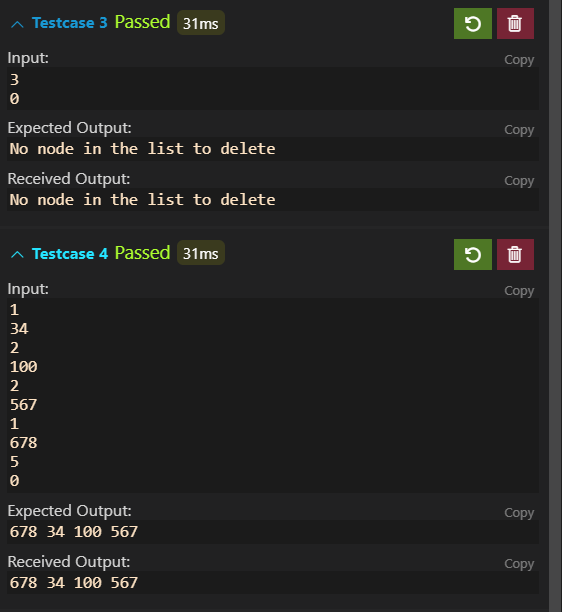
    return 0;

}

**Input & Output:**

****

****

****

**Conclusion:** From this program I have learned to create the Doubly Linked List from scratch using class in c++.

1. **Aim:** Instructors have given one problem to students. Students have to solve and implement the program in maximum 60 minutes. After contest, Instructor has the submission time (in minutes) in which every student has submitted the correct solution.

Help the instructor to find the top 2 students who have submitted correct problem fastest. Also, instructor wants to know the time taken by a particular student based on his/her roll number.

Input for this program is roll number, roll\_no and time to submit the solution in minutes, submissiontime for n students.

Explanation:

Input will be: n=5 (number of students)

roll\_no: 1,4,5,7,10

submissiontime= 34,60,42,35,21

1 (for Bubble Sort)

2 (for Selection Sort)

Output can be:

case 1: Bubble Sort :

21,34,35,42,60

No of swaps : 7

case 2: Selection Sort :

21,34,35,42,60

No of swaps : 3

Top 2 students : roll no : 10 and 1

**Program:**

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

void bubble\_sort(int \*a, int \*b, int n)

{

    int comp = 0;

    int swaps = 0;

    int flag = false;

    for (int i = 0; i < n; i++)

    {

        flag = false;

        for (int j = 0; j < n - i - 1; j++)

        {

            comp++;

            if (b[j] > b[j + 1])

            {

                flag = true;

                swap(a[j], a[j + 1]);

                swap(b[j], b[j + 1]);

                swaps++;

            }

        }

        if (flag == false)

            break;

    }

    for (int i = 0; i < n; i++)

    {

        cout << a[i] << " " << b[i] << endl;

    }

    // comp = n\*(n-1)/2;

    // cout<<n\*(n-1)/2;

    cout << comp << endl

         << swaps << endl

         << a[0] << "," << a[1];

}

void selection\_sort(int \*a, int \*b, int n)

{

    int comp = 0;

    int swaps = 0;

    for (int i = 0; i < n; i++)

    {

        int mn = i;

        for (int j = i + 1; j < n; j++)

        {

            if (b[j] < b[mn])

                mn = j;

            comp++;

        }

        if (mn != i)

        {

            swaps++;

            swap(a[i], a[mn]);

            swap(b[i], b[mn]);

        }

    }

    for (int i = 0; i < n; i++)

    {

        cout << a[i] << " " << b[i] << endl;

    }

    cout << comp << endl

         << swaps << endl

         << a[0] << "," << a[1];

}

int main()

{

    /\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/ int n;

    cin >> n;

    int a[n];

    int b[n];

    for (int i = 0; i < n; i++)

    {

        cin >> a[i];

        cin >> b[i];

    }

    int which;

    cin >> which;

    // cout<<which<<endl;

    if (which == 1)

        bubble\_sort(a, b, n);

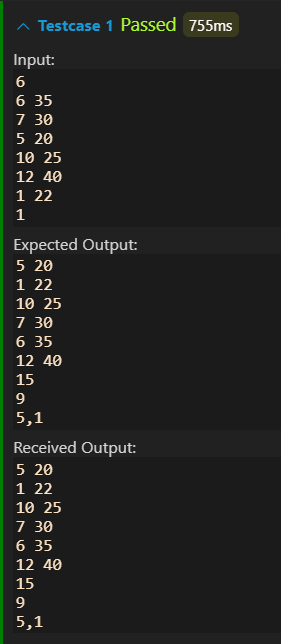
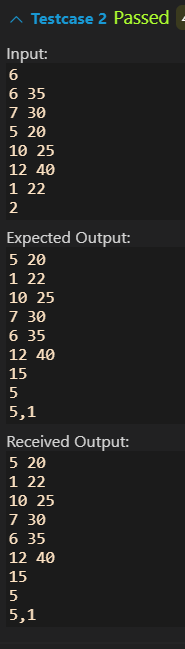
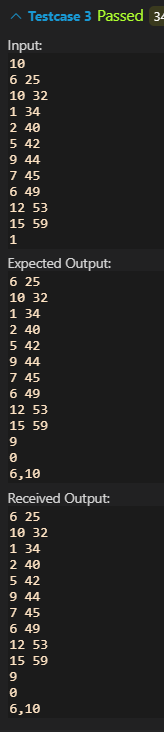
    else

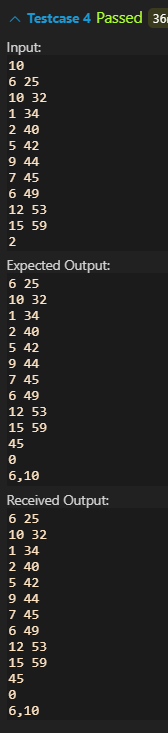
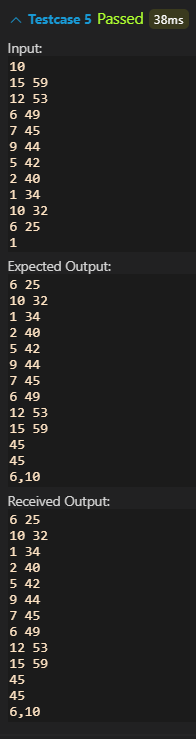
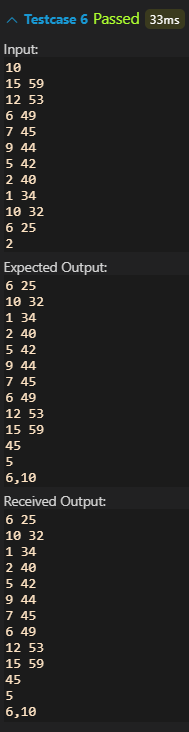
        selection\_sort(a, b, n);

    return 0;

}

**Input &Output:**

**  **

**  **

**Conclusion:** From this program I have learned to create the basic sorting algorithms such as bubble sort, selection sort, etc.