

**DSA (Data Structure and Algorithms) Lab**

**LAB REPORT # 10**

**Semester**: 3rdSemester

**Section**: C

**Submitted To:**

**Abdullah Shahrose**

**Submitted By:**

**Name**: Faisal Khan

**Roll No**: 22-CS-039

**Task 1:**

#include <iostream>

void swap(int &a, int &b) {

    int temp = a;

    a = b;

    b = temp;

}

void bubbleSort(int arr[], int n) {

    for (int i = 0; i < n - 1; ++i) {

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

            if (arr[j] > arr[j + 1]) {

                // Swap arr[j] and arr[j+1]

                swap(arr[j], arr[j + 1]);

            }

        }

    }

}

void printArray(const int arr[], int size) {

    for (int i = 0; i < size; i++)

        std::cout << arr[i] << " ";

    std::cout << std::endl;

}

int main() {

    const int size = 7;

    int arr[size] = {64, 34, 25, 12, 22, 11, 90};

    std::cout << "Unsorted array: ";

    printArray(arr, size);

    bubbleSort(arr, size);

    std::cout << "Sorted array: ";

    printArray(arr, size);

    return 0;

}

**Output:**

****

**Task 2:**

#include <iostream>

class Node

{

public:

    int data;

    Node \*next;

    Node(int value) : data(value), next(nullptr) {}

};

void swap(Node \*a, Node \*b)

{

    int temp = a->data;

    a->data = b->data;

    b->data = temp;

}

void bubbleSort(Node \*start)

{

    int swapped;

    Node \*ptr1;

    Node \*lptr = nullptr;

    // Checking for empty list

    if (start == nullptr)

        return;

    do

    {

        swapped = 0;

        ptr1 = start;

        while (ptr1->next != lptr)

        {

            if (ptr1->data > ptr1->next->data)

            {

                swap(ptr1, ptr1->next);

                swapped = 1;

            }

            ptr1 = ptr1->next;

        }

        lptr = ptr1;

    } while (swapped);

}

void printList(Node \*node)

{

    while (node != nullptr)

    {

        std::cout << node->data << " ";

        node = node->next;

    }

    std::cout << std::endl;

}

int main()

{

    Node \*head = new Node(64);

    head->next = new Node(34);

    head->next->next = new Node(25);

    head->next->next->next = new Node(12);

    head->next->next->next->next = new Node(22);

    head->next->next->next->next->next = new Node(11);

    head->next->next->next->next->next->next = new Node(90);

    std::cout << "Unsorted list: ";

    printList(head);

    bubbleSort(head);

    std::cout << "Sorted list: ";

    printList(head);

    return 0;

}

**Output:**



**Task 3:**

#include <iostream>

class Node

{

public:

    int data;

    Node \*next;

    Node \*prev;

    Node(int value) : data(value), next(nullptr), prev(nullptr) {}

};

void swap(Node \*a, Node \*b)

{

    int temp = a->data;

    a->data = b->data;

    b->data = temp;

}

void bubbleSort(Node \*start)

{

    int swapped;

    Node \*ptr1;

    Node \*lptr = nullptr;

    if (start == nullptr)

        return;

    do

    {

        swapped = 0;

        ptr1 = start;

        while (ptr1->next != lptr)

        {

            if (ptr1->data > ptr1->next->data)

            {

                swap(ptr1, ptr1->next);

                swapped = 1;

            }

            ptr1 = ptr1->next;

        }

        lptr = ptr1;

    } while (swapped);

}

void printList(Node \*node)

{

    while (node != nullptr)

    {

        std::cout << node->data << " ";

        node = node->next;

    }

    std::cout << std::endl;

}

int main()

{

    Node \*head = new Node(64);

    head->next = new Node(34);

    head->next->prev = head;

    head->next->next = new Node(25);

    head->next->next->prev = head->next;

    head->next->next->next = new Node(12);

    head->next->next->next->prev = head->next->next;

    head->next->next->next->next = new Node(22);

    head->next->next->next->next->prev = head->next->next->next;

    head->next->next->next->next->next = new Node(11);

    head->next->next->next->next->next->prev = head->next->next->next->next;

    head->next->next->next->next->next->next = new Node(90);

    head->next->next->next->next->next->next->prev = head->next->next->next->next->next;

    std::cout << "Unsorted list: ";

    printList(head);

    bubbleSort(head);

    std::cout << "Sorted list: ";

    printList(head);

    return 0;

}

**Output:**

****