```
#include <iostream>
#include <vector>
using namespace std;

/* ================= LINKED LIST ================= */

struct Node {
    int data;
    Node* next;
};

Node* head = NULL;

void insertNode(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
    cout << "Node inserted successfully.\n";
}

void deleteNode(int value) {
    Node* temp = head;
    Node* prev = NULL;

    if (temp != NULL && temp->data == value) {
        head = temp->next;
        delete temp;
        cout << "Node deleted successfully.\n";
        return;
    }

    while (temp != NULL && temp->data != value) {
        prev = temp;
        temp = temp->next;
```

```
    }

    if (temp == NULL) {
        cout << "Value not found.\n";
        return;
    }

    prev->next = temp->next;
    delete temp;
    cout << "Node deleted successfully.\n";
}

void displayList() {
    if (head == NULL) {
        cout << "Linked List is empty.\n";
        return;
    }

    Node* temp = head;
    cout << "Linked List: ";
    while (temp != NULL) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

/* ================= MERGE SORT ================= */

void merge(vector<int>& arr, int left, int mid, int right) {
    vector<int> temp;
    int i = left, j = mid + 1;

    while (i <= mid && j <= right) {
        if (arr[i] <= arr[j])
            temp.push_back(arr[i++]);
        else
            temp.push_back(arr[j++]);
    }

    while (i <= mid) temp.push_back(arr[i++]);
    while (j <= right) temp.push_back(arr[j++]);
```

```
    for (int k = 0; k < temp.size(); k++)
        arr[left + k] = temp[k];
}

void mergeSort(vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

/* ================= QUICK SORT ================= */

int partition(vector<int>& arr, int low, int high) {
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return i + 1;
}

void quickSort(vector<int>& arr, int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

/* ================= BINARY SEARCH ================= */

int binarySearch(vector<int>& arr, int key) {
    int low = 0, high = arr.size() - 1;

    while (low <= high) {
```

```
        int mid = (low + high) / 2;

        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}

/* ================= MAIN MENU ================= */

int main() {
    vector<int> arr;
    int choice, value, n, key;

    do {
        cout << "\n====== MENU ======\n";
        cout << "1. Insert into Linked List\n";
        cout << "2. Delete from Linked List\n";
        cout << "3. Display Linked List\n";
        cout << "4. Enter Array Elements\n";
        cout << "5. Merge Sort\n";
        cout << "6. Quick Sort\n";
        cout << "7. Binary Search\n";
        cout << "8. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter value: ";
            cin >> value;
            insertNode(value);
            break;

        case 2:
            cout << "Enter value to delete: ";
            cin >> value;
            deleteNode(value);
            break;
```

```
    case 3:
        displayList();
        break;

    case 4:
        cout << "Enter number of elements: ";
        cin >> n;
        arr.clear();
        for (int i = 0; i < n; i++) {
            cin >> value;
            arr.push_back(value);
        }
        break;

    case 5:
        mergeSort(arr, 0, arr.size() - 1);
        cout << "Array sorted using Merge Sort.\n";
        break;

    case 6:
        quickSort(arr, 0, arr.size() - 1);
        cout << "Array sorted using Quick Sort.\n";
        break;

    case 7:
        cout << "Enter element to search: ";
        cin >> key;
        value = binarySearch(arr, key);
        if (value != -1)
            cout << "Element found at index " << value << endl;
        else
            cout << "Element not found.\n";
        break;

    case 8:
        cout << "Exiting program.\n";
        break;

    default:
        cout << "Invalid choice.\n";
    }
} while (choice != 8);
```

```
    return 0;
}
```