



ChatGPT ▾

Share



ChatGPT

Hot Mods

Explore GPTs

Today

Linked List Operations

Yesterday

C++ Linked List Programs ...

Sci-Fi Group Names

Coding Student Illustration Requ...

DSA Program Solutions

Previous 7 Days

Pointers in C

Barcode Information Inquiry

Infinite Recursion Explanation

C++ Function Swap Correction

1. Write a program to create an array of n elements and then display all the elements of the list.
2. Write a program to find the largest number from a given list of integers.
3. Write a program to calculate the roots of the quadratic equation $ax^2 + bx + c = 0$ where a, b and c are known.
4. Write a program to create an array of n elements and then separately write the odd and even elements of the list.
5. Write a program to create an array of n elements and then insert an element to the list.
6. Write a program to create an array of n elements and then delete an element from the list.
7. Write a program to sort n numbers using Bubble Sort algorithm.
8. Write a program to search an element from a list of n numbers using Linear Search algorithm.
9. Write a program to search an element from a list of n numbers using Binary Search algorithm.

solve this with C++ using linked list with sample input and sample output like cin cout.

remember, give me ↴ way and simple way.

1. Create a linked list of n elements and display the elements.

cpp

Copy code

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

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

void display(Node* head) {
    cout << "Elements in the linked list: ";
    while (head != nullptr) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    int n, value;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    Node* temp = nullptr;
    Node* tail = nullptr;

    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;

        temp = new Node{value, nullptr};
        if (head == nullptr) {
            head = temp;
            tail = temp;
        } else {
            tail->next = temp;
            tail = temp;
        }
    }

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

Sample Input:

mathematica

Copy code

```
Enter number of elements: 3
Enter value: 5
Enter value: 10
Enter value: 15
```

Sample Output:

yaml

 Copy code

```
Elements in the linked list: 5 10 15
```



2. Find the largest number from a given linked list of integers.

cpp

 Copy code

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

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

int findLargest(Node* head) {
    if (head == nullptr) return -1; // Assuming -1 as error value for empty list
    int largest = head->data;
    head = head->next;
    while (head != nullptr) {
        if (head->data > largest) {
            largest = head->data;
        }
        head = head->next;
    }
    return largest;
}

int main() {
    int n, value;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    Node* temp = nullptr;
    Node* tail = nullptr;

    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;

        temp = new Node{value, nullptr};
        if (head == nullptr) {
            head = temp;
            tail = temp;
        } else {
            tail->next = temp;
            tail = temp;
        }
    }

    cout << "The largest element is: " << findLargest(head) << endl;
    return 0;
}
```

Sample Input:

mathematica

 Copy code

```
Enter number of elements: 4
Enter value: 7
Enter value: 3
Enter value: 15
Enter value: 9
```

Sample Output:

csharp

 Copy code

```
The largest element is: 15
```

3. Calculate the roots of a quadratic equation $ax^2 + bx + c = 0$.

```
cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    double a, b, c;
    cout << "Enter coefficients a, b, and c: ";
    cin >> a >> b >> c;

    if (a == 0) {
        cout << "This is not a quadratic equation." << endl;
        return 0;
    }

    double discriminant = b * b - 4 * a * c;
    if (discriminant > 0) {
        double root1 = (-b + sqrt(discriminant)) / (2 * a);
        double root2 = (-b - sqrt(discriminant)) / (2 * a);
        cout << "Roots are real and different." << endl;
        cout << "Root 1: " << root1 << endl;
        cout << "Root 2: " << root2 << endl;
    } else if (discriminant == 0) {
        double root = -b / (2 * a);
        cout << "Roots are real and the same." << endl;
        cout << "Root: " << root << endl;
    } else {
        double realPart = -b / (2 * a);
        double imaginaryPart = sqrt(-discriminant) / (2 * a);
        cout << "Roots are complex and different." << endl;
        cout << "Root 1: " << realPart << " + " << imaginaryPart << "i" << endl;
        cout << "Root 2: " << realPart << " - " << imaginaryPart << "i" << endl;
    }
}

return 0;
}
```

Sample Input:

Enter coefficients a, b, and c: `1 -3 2`

Sample Output:

Let me know if you'd like the remaining solutions or any additional modifications!



i give you 9



Here are the remaining solutions for the linked list problems you requested. I'll give simple and easy-to-understand solutions for the remaining tasks using linked lists.

4. Create a linked list of n elements and then separately write the odd and even elements of the list.

```

        Node* next;
    };

    void displayOddEven(Node* head) {
        cout << "Odd elements: ";
        while (head != nullptr) {
            if (head->data % 2 != 0) {
                cout << head->data << " ";
            }
            head = head->next;
        }
        cout << endl;
    }

    cout << "Even elements: ";
    head = head; // Reset to head for even display
    while (head != nullptr) {
        if (head->data % 2 == 0) {
            cout << head->data << " ";
        }
        head = head->next;
    }
    cout << endl;
}

int main() {
    int n, value;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    Node* temp = nullptr;
    Node* tail = nullptr;

    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;

        temp = new Node{value, nullptr};
        if (head == nullptr) {
            head = temp;
            tail = temp;
        } else {
            tail->next = temp;
            tail = temp;
        }
    }

    displayOddEven(head);
    return 0;
}

```

Sample Input:

mathematica

Copy code

```

Enter number of elements: 5
Enter value: 7
Enter value: 2
Enter value: 4
Enter value: 9
Enter value: 10

```

Sample Output:

yaml

Copy code

```

Odd elements: 7 9
Even elements: 2 4 10

```

5. Create a linked list of n elements and then insert an element to the list.

cpp

Copy code

```

#include <iostream>
using namespace std;

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

```

```

void insert(Node*& head, int value) {
    Node* newNode = new Node{value, nullptr};
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void display(Node* head) {
    while (head != nullptr) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    int n, value, newValue;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;
        insert(head, value);
    }

    cout << "Enter a new value to insert: ";
    cin >> newValue;
    insert(head, newValue);

    cout << "Updated list: ";
    display(head);
    return 0;
}

```

Sample Input:

mathematica

Copy code

```

Enter number of elements: 3
Enter value: 5
Enter value: 10
Enter value: 15
Enter a new value to insert: 20

```

Sample Output:

yaml

Copy code

```
Updated list: 5 10 15 20
```

6. Create a linked list of n elements and then delete an element from the list.

cpp

Copy code

```

#include <iostream>
using namespace std;

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

void deleteElement(Node*& head, int value) {
    if (head == nullptr) return;

    if (head->data == value) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }
}

```

```

Node* temp = head;
while (temp->next != nullptr && temp->next->data != value) {
    temp = temp->next;
}

if (temp->next != nullptr) {
    Node* toDelete = temp->next;
    temp->next = temp->next->next;
    delete toDelete;
}
}

void display(Node* head) {
    while (head != nullptr) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    int n, value, deleteValue;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;
        Node* newNode = new Node{value, nullptr};
        if (head == nullptr) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    cout << "Enter value to delete: ";
    cin >> deleteValue;
    deleteElement(head, deleteValue);

    cout << "Updated list: ";
    display(head);
    return 0;
}

```

Sample Input:

mathematica

Copy code

```

Enter number of elements: 4
Enter value: 5
Enter value: 10
Enter value: 15
Enter value: 20
Enter value to delete: 15

```

Sample Output:

yaml

Copy code

```
Updated list: 5 10 20
```

7. Sort n numbers using the Bubble Sort algorithm.

cpp

Copy code

```

#include <iostream>
using namespace std;

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

```

```

void bubbleSort(Node* head) {
    if (head == nullptr) return;
    bool swapped;
    do {
        swapped = false;
        Node* temp = head;
        while (temp != nullptr && temp->next != nullptr) {
            if (temp->data > temp->next->data) {
                swap(temp->data, temp->next->data);
                swapped = true;
            }
            temp = temp->next;
        }
    } while (swapped);
}

void display(Node* head) {
    while (head != nullptr) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    int n, value;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;
        Node* newNode = new Node{value, nullptr};
        if (head == nullptr) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    bubbleSort(head);
    cout << "Sorted list: ";
    display(head);
    return 0;
}

```

Sample Input:

mathematica

 Copy code

```

Enter number of elements: 4
Enter value: 20
Enter value: 10
Enter value: 30
Enter value: 5

```

Sample Output:

yaml

 Copy code

```

sorted list: 5 10 20 30

```

8. Search an element from a linked list using the Linear Search algorithm.

cpp

 Copy code

```

#include <iostream>
using namespace std;

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

bool linearSearch(Node* head, int value) {

```

```

        while (head != nullptr) {
            if (head->data == value) return true;
            head = head->next;
        }
        return false;
    }

int main() {
    int n, value, searchValue;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;
        Node* newNode = new Node{value, nullptr};
        if (head == nullptr) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    cout << "Enter value to search: ";
    cin >> searchValue;

    if (linearSearch(head, searchValue)) ↓
        cout << "Element found!" << endl;
    } else {
        cout << "Element not found!" << endl;
    }

    return 0;
}

```

Sample Input:

mathematica

Copy code

```

Enter number of elements: 5
Enter value: 5
Enter value: 10
Enter value: 20
Enter value: 25
Enter value: 30
Enter value to search: 20

```

Sample Output:

mathematica

Copy code

```

Element found!

```

9. Search an element from a linked list using the Binary Search algorithm.

Note: Binary search requires the list to be sorted. Here's the solution assuming the list is already sorted.

cpp

Copy code

```

#include <iostream>
using namespace std;

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

bool binarySearch(Node* head, int value) ↓
Node* left = head;
Node* right = nullptr;

// Get last node
right = head;
while (right != nullptr && right->next != nullptr) {
    right = right->next;
}

if (right->data == value) return true;
return false;
}

```

```
    right = right->next;
}

while (left != right) {
    Node* mid = left;
    Node* slow = left;
    Node* fast = left;

    // Move slow to the middle
    while (fast != nullptr && fast->next != right) {
        slow = slow->next;
        fast = fast->next;
    }

    if (mid->data == value) return true;
    if (mid->data < value) {
        left = slow->next;
    } else {
        right = slow;
    }
}

return false;
}

int main() {
    int n, value, searchValue;
    cout << "Enter number of elements: ";
    cin >> n;

    Node* head = nullptr;
    for (int i = 0; i < n; ++i) {
        cout << "Enter value: ";
        cin >> value;
        Node* newNode = new Node(value, nullptr);
        if (head == nullptr) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    cout << "Enter value to search: ";
    cin >> searchValue;

    if (binarySearch(head, searchValue)) {
        cout << "Element found!" << endl;
    } else {
        cout << "Element not found!" << endl;
    }

    return 0;
}
```

Sample Input:

Sample Output:

Feel free to ask if you'd like any modifications or more explanations!

C/C++ Function Swap Correction

 **Upgrade plan**
More access to the best models

 Message ChatGPT



ChatGPT can make mistakes. Check important info.

