1- Write a C++ program that demonstrates the use of dynamic memory allocation for a linked list of members. Each member has a name and payment information. Implement a linked list using a class, where each node represents a member. The program should allow users to add members and display the details of all members.

اكتب برنامج C++ يوضح استخدام تخصيص الذاكرة الديناميكية لقائمة مرتبطة من الأعضاء. كل عضو لديه اسم ومعلومات الدفع. قم بتنفيذ قائمة مرتبطة باستخدام فئة، حيث تمثل كل عقدة عضوًا. يجب أن يسمح البرنامج للمستخدمين بإضافة أعضاء وعرض تفاصيل جميع الأعضاء.

Input & Output

```
New Member created
Enter name: ahmed
Enter payment: 3000
1) Add
2) Show
3) Exit
Enter a number: 1
New Member created
Enter name: aly
Enter payment: 2000
1) Add
2) Show
3) Exit
Enter a number: 1
New Member created
Enter name: mena
Enter payment: 5000
1) Add
2) Show
3) Exit
Enter a number: 2
--------------
Name: ahmed
Payment: 3000
--------------
Name: aly
Payment: 2000
--------------
Name: mena
Payment: 5000
1) Add
2) Show
3) Exit
Enter a number: 3
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Member {
private:
    string name;
    int payment;
    Member* next;

public:
    Member() {
        cout << "New Member created" << endl;
    }

    ~Member() {
        cout << "Member Deleted" << endl;
    }

    void getdata() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter payment: ";
        cin >> payment;
        next = NULL;
    }

    void print() {
        cout << "--------------" << endl;
        cout << "Name: " << name << endl;
        cout << "Payment: " << payment << endl;
    }

    friend Member* add(Member* g);
    friend void show(Member* g);
};

Member* add(Member* g) {
    if (g == NULL) {
        g = new Member;
        g->getdata();
        return g;
    }
    g->next = add(g->next);
    return g;
}

void show(Member* g) {
    while (g != NULL) {
        g->print();
        g = g->next;
    }
}

int main() {
    int choice;
    Member* head = NULL;

    do {
        cout << "1) Add\n2) Show\n3) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                head = add(head);
                break;
            case 2:
                show(head);
                break;
        }
    } while (choice != 3);

    return 0;
}
```

2- Write a C++ program that demonstrates the use of dynamic memory allocation for a linked list. Implement a linked list using a class (Node), where each node has an integer data value. The program should allow users to add nodes and display the linked list.

اكتب برنامج C++ يوضح استخدام تخصيص الذاكرة الديناميكية لقائمة مرتبطة. تنفيذ قائمة مرتبطة باستخدام فئة (عقدة)، حيث تحتوي كل عقدة على قيمة بيانات عددية. يجب أن يسمح البرنامج للمستخدمين بإضافة العقد وعرض القائمة المرتبطة.

Input & Output

```
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 42
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 43
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 44
1) Add Node
2) Display List
3) Exit
Enter a number: 2
Linked List: 42 43 44
1) Add Node
2) Display List
3) Exit
Enter a number: 3
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

void displayList(Node* head) {
    cout << "Linked List: ";
    while (head != NULL) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Display List\n3) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                displayList(head);
                break;
        }
    } while (choice != 3);

    return 0;
}
```

3- Write a C++ program that demonstrates the use of dynamic memory allocation for a linked list using classes. Create a LinkedList class that contains a Node class. Each Node has an integer data value, and the LinkedList class manages the linked list. The program should allow users to add nodes and display the linked list. Ensure that the program properly deallocates memory when the linked list and nodes go out of scope.

اكتب برنامج C++ يوضح استخدام تخصيص الذاكرة الديناميكية لقائمة مرتبطة باستخدام الفئات. قم بإنشاء فئة LinkedList التي تحتوي على فئة Node. تحتوي كل عقدة على قيمة بيانات عددية، وتدير فئة LinkedList القائمة المرتبطة. يجب أن يسمح البرنامج للمستخدمين بإضافة العقد وعرض القائمة المرتبطة. تأكد من أن البرنامج يقوم بإلغاء تخصيص الذاكرة بشكل صحيح عندما تخرج القائمة والعقد المرتبطة عن النطاق.

Output

```
Linked List created
Node created with data: 10
Node created with data: 20
Node created with data: 30
Linked List: 10 20 30
Node deleted with data: 10
Node deleted with data: 20
Node deleted with data: 30
Linked List deleted
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = NULL;
        cout << "Linked List created" << endl;
    }

    ~LinkedList() {
        while (head != NULL) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
        cout << "Linked List deleted" << endl;
    }

    void addNode(int value) {
        if (head == NULL) {
            head = new Node(value);
        } else {
            Node* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = new Node(value);
        }
    }

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

int main() {
    LinkedList myList;

    myList.addNode(10);
    myList.addNode(20);
    myList.addNode(30);

    myList.displayList();

    // Destructor of LinkedList class is automatically called when the program exits
    return 0;
}
```

4- Write a program that creates a linked list of integers. The program should provide the following options:

Add Node: Adds a new node to the end of the linked list. The user will enter the integer value for the new node.

Display List: Displays the linked list.

Exit: Exits the program.

Implement the necessary functions for adding a node and displaying the list. Ensure proper memory management.

اكتب برنامجا يقوم بإنشاء قائمة مرتبطة من الأعداد الصحيحة. يجب أن يوفر البرنامج الخيارات التالية:

إضافة عقدة: إضافة عقدة جديدة إلى نهاية القائمة المرتبطة. سيقوم المستخدم بإدخال القيمة الصحيحة للعقدة الجديدة.

عرض القائمة: يعرض القائمة المرتبطة.

خروج: الخروج من البرنامج.

قم بتنفيذ الوظائف اللازمة لإضافة عقدة وعرض القائمة. ضمان الإدارة السليمة للذاكرة.

Input & Output

```
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 4
Node created with data: 4
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 5
Node created with data: 5
1) Add Node
2) Display List
3) Exit
Enter a number: 2
Linked List: 1 2 3 4 5
1) Add Node
2) Display List
3) Exit
Enter a number: 3
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

void displayList(Node* head) {
    cout << "Linked List: ";
    while (head != NULL) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Display List\n3) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                displayList(head);
                break;
        }
    } while (choice != 3);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```

5- Enhance previous the linked list program by adding an option to calculate the sum of all numbers in the linked list.

Add Node: Adds a new node to the end of the linked list. The user will enter the integer value for the new node.

Display Even Numbers: Displays the even numbers in the linked list.

Calculate Sum: Calculates and displays the sum of all numbers in the linked list.

Exit: Exits the program.

قم بتحسين برنامج القائمة المرتبطة عن طريق إضافة خيار لحساب مجموع جميع الأرقام في القائمة المرتبطة.

إضافة عقدة: إضافة عقدة جديدة إلى نهاية القائمة المرتبطة. سيقوم المستخدم بإدخال القيمة الصحيحة للعقدة الجديدة.

عرض الأرقام الزوجية: يعرض الأرقام الزوجية في القائمة المرتبطة.

حساب المجموع: حساب وعرض مجموع جميع الأرقام في القائمة المرتبطة.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 1
Enter a value: 4
Node created with data: 4
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 2
Even Numbers in Linked List: 2 4
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 3
Sum of all numbers: 10
1) Add Node
2) Display Even Numbers
3) Calculate Sum
4) Exit
Enter a number: 4
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

void displayEvenNumbers(Node* head) {
    cout << "Even Numbers in Linked List: ";
    while (head != NULL) {
        if (head->data % 2 == 0) {
            cout << head->data << " ";
        }
        head = head->next;
    }
    cout << endl;
}

int calculateSum(Node* head) {
    int sum = 0;
    while (head != NULL) {
        sum += head->data;
        head = head->next;
    }
    return sum;
}

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Display Even Numbers\n3) Calculate Sum\n4) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                displayEvenNumbers(head);
                break;
            case 3:
                cout << "Sum of all numbers: " << calculateSum(head) << endl;
                break;
        }
    } while (choice != 4);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```

6- Modify the linked list program by adding an option to reverse the linked list.

Add Node: Adds a new node to the end of the linked list. The user will enter the integer value for the new node.

Display List: Displays the current linked list.

Reverse List: Reverses the linked list.

Exit: Exits the program.

تعديل برنامج القائمة المرتبطة عن طريق إضافة خيار لعكس القائمة المرتبطة.

إضافة عقدة: إضافة عقدة جديدة إلى نهاية القائمة المرتبطة. سيقوم المستخدم بإدخال القيمة الصحيحة للعقدة الجديدة.

قائمة العرض: تعرض القائمة المرتبطة الحالية.

القائمة العكسية: لعكس القائمة المرتبطة.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 1
Enter a value: 4
Node created with data: 4
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 2
1 2 3 4
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 3
List Reversed!
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 2
4 3 2 1
1) Add Node
2) Display List
3) Reverse List
4) Exit
Enter a number: 4
```


Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

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

Node* reverseList(Node* head) {
    Node* prev = NULL;
    Node* current = head;
    Node* nextNode = NULL;

    while (current != NULL) {
        nextNode = current->next;
        current->next = prev;
        prev = current;
        current = nextNode;
    }

    return prev;
}

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Display List\n3) Reverse List\n4) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                displayList(head);
                break;
            case 3:
                head = reverseList(head);
                cout << "List Reversed!" << endl;
                break;
        }
    } while (choice != 4);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```

7- Extend the linked list program to include the option to calculate the sum of all odd numbers in the linked list.

Add Node: Adds a new node to the end of the linked list. The user will enter the integer value for the new node.

Display List: Displays the current linked list.

Reverse List: Reverses the linked list.

Sum Odd Numbers: Calculates and displays the sum of all odd numbers in the linked list.

Exit: Exits the program.

قم بتوسيع برنامج القائمة المرتبطة ليشمل خيار حساب مجموع جميع الأرقام الفردية في القائمة المرتبطة.

إضافة عقدة: إضافة عقدة جديدة إلى نهاية القائمة المرتبطة. سيقوم المستخدم بإدخال القيمة الصحيحة للعقدة الجديدة.

قائمة العرض: تعرض القائمة المرتبطة الحالية.

القائمة العكسية: لعكس القائمة المرتبطة.

مجموع الأرقام الفردية: يحسب ويعرض مجموع جميع الأرقام الفردية في القائمة المرتبطة.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 1
Enter a value: 4
Node created with data: 4
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 1
Enter a value: 5
Node created with data: 5
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 2
1 2 3 4 5
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 3
List Reversed!
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 2
5 4 3 2 1
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 4
Sum of Odd Numbers: 9
1) Add Node
2) Display List
3) Reverse List
4) Sum Odd Numbers
5) Exit
Enter a number: 5
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

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

Node* reverseList(Node* head) {
    Node* prev = NULL;
    Node* current = head;
    Node* nextNode = NULL;

    while (current != NULL) {
        nextNode = current->next;
        current->next = prev;
        prev = current;
        current = nextNode;
    }

    return prev;
}

int sumOddNumbers(Node* head) {
    int sum = 0;
    while (head != NULL) {
        if (head->data % 2 != 0) {
            sum += head->data;
        }
        head = head->next;
    }
    return sum;
}

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Display List\n3) Reverse List\n4) Sum Odd Numbers\n5) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                displayList(head);
                break;
            case 3:
                head = reverseList(head);
                cout << "List Reversed!" << endl;
                break;
            case 4:
                cout << "Sum of Odd Numbers: " << sumOddNumbers(head) << endl;
                break;
        }
    } while (choice != 5);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```

8- Write a program to include the option to display all negative numbers in the linked list.

Add Node: Adds a new node to the end of the linked list. The user will enter the integer value for the new node.

Display Negative Numbers: Displays all negative numbers in the linked list.

Exit: Exits the program.

كتابة برنامج يتضمن خيار عرض جميع الأرقام السالبة في القائمة المرتبطة.

إضافة عقدة: إضافة عقدة جديدة إلى نهاية القائمة المرتبطة. سيقوم المستخدم بإدخال القيمة الصحيحة للعقدة الجديدة.

عرض الأرقام السالبة: يعرض كافة الأرقام السالبة في القائمة المرتبطة.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Display Negative Numbers
3) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Display Negative Numbers
3) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Display Negative Numbers
3) Exit
Enter a number: 1
Enter a value: -5
Node created with data: -5
1) Add Node
2) Display Negative Numbers
3) Exit
Enter a number: 1
Enter a value: -8
Node created with data: -8
1) Add Node
2) Display Negative Numbers
3) Exit
Enter a number: 2
Negative Numbers: -5 -8
1) Add Node
2) Display Negative Numbers
3) Exit
Enter a number: 3
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

void displayNegativeNumbers(Node* head) {
    cout << "Negative Numbers: ";
    while (head != NULL) {
        if (head->data < 0) {
            cout << head->data << " ";
        }
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Display Negative Numbers\n3) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                displayNegativeNumbers(head);
                break;
        }
    } while (choice != 3);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```

9- Write a program to include the option to display all numbers smaller than a given threshold.

Add Node: Adds a new node to the end of the linked list. The user will enter the integer value for the new node.

Display Smaller Numbers: Displays all numbers in the linked list that are smaller than a specified threshold. The user will enter the threshold value.

Exit: Exits the program.

اكتب برنامجًا يتضمن خيار عرض جميع الأرقام الأصغر من حد معين.

إضافة عقدة: إضافة عقدة جديدة إلى نهاية القائمة المرتبطة. سيقوم المستخدم بإدخال القيمة الصحيحة للعقدة الجديدة.

عرض أرقام أصغر: يعرض جميع الأرقام الموجودة في القائمة المرتبطة والتي تكون أصغر من الحد المحدد. سيقوم المستخدم بإدخال قيمة العتبة.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Display Smaller Numbers
3) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Display Smaller Numbers
3) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Display Smaller Numbers
3) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Display Smaller Numbers
3) Exit
Enter a number: 1
Enter a value: 0
Node created with data: 0
1) Add Node
2) Display Smaller Numbers
3) Exit
Enter a number: 2
Enter a threshold value: 2
Numbers Smaller Than 2: 1 0
1) Add Node
2) Display Smaller Numbers
3) Exit
Enter a number: 3
```


Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

void displaySmallerNumbers(Node* head, int threshold) {
    cout << "Numbers Smaller Than " << threshold << ": ";
    while (head != NULL) {
        if (head->data < threshold) {
            cout << head->data << " ";
        }
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = NULL;
    int choice, value, threshold;

    do {
        cout << "1) Add Node\n2) Display Smaller Numbers\n3) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                cout << "Enter a threshold value: ";
                cin >> threshold;
                displaySmallerNumbers(head, threshold);
                break;
        }
    } while (choice != 3);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```

10- create a C++ program to find the largest number in a linked list.

إنشاء برنامج C++ للعثور على أكبر رقم في القائمة المرتبطة.

Input & Output

```
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 1
Enter a value: 5
Node created with data: 5
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 2
Largest Number: 5
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 3
List: 2 1 5 3
1) Add Node
2) Find Largest Number
3) Display List
4) Exit
Enter a number: 4
```


Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = NULL;
        cout << "Node created with data: " << data << endl;
    }

    ~Node() {
        cout << "Node deleted with data: " << data << endl;
    }
};

Node* addNode(Node* head, int value) {
    if (head == NULL) {
        head = new Node(value);
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new Node(value);
    }
    return head;
}

int findLargest(Node* head) {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return -1;
    }

    int largest = head->data;
    Node* current = head->next;

    while (current != NULL) {
        if (current->data > largest) {
            largest = current->data;
        }
        current = current->next;
    }

    return largest;
}

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

int main() {
    Node* head = NULL;
    int choice, value;

    do {
        cout << "1) Add Node\n2) Find Largest Number\n3) Display List\n4) Exit\n";
        cout << "Enter a number: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value: ";
                cin >> value;
                head = addNode(head, value);
                break;
            case 2:
                cout << "Largest Number: " << findLargest(head) << endl;
                break;
            case 3:
                cout << "List: ";
                displayList(head);
                break;
        }
    } while (choice != 4);

    // The destructor of each node is automatically called when the program exits
    return 0;
}
```