1- Create a C++ program to print the elements of a linked list.

إنشاء برنامج C++ لطباعة عناصر القائمة المرتبطة.

Input & Output

```
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 2
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 3
1) Add Node
2) Display List
3) Exit
Enter a number: 1
Enter a value: 4
1) Add Node
2) Display List
3) Exit
Enter a number: 2
Linked List: 2 3 4
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) {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }

    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;
}
```

2- Create a C++ program to count the nodes in a linked list.

قم بإنشاء برنامج C++ لحساب العقد في القائمة المرتبطة.

Input & Output

```
1) Add Node
2) Count Nodes
3) Exit
Enter a number: 1
Enter a value: 2
1) Add Node
2) Count Nodes
3) Exit
Enter a number: 1
Enter a value: 3
1) Add Node
2) Count Nodes
3) Exit
Enter a number: 1
Enter a value: 4
1) Add Node
2) Count Nodes
3) Exit
Enter a number: 2
Number of Nodes: 3
1) Add Node
2) Count Nodes
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() {
        // Destructor
    }
};

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 countNodes(Node* head) {
    int count = 0;
    while (head != NULL) {
        count++;
        head = head->next;
    }
    return count;
}

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

    do {
        cout << "1) Add Node\n2) Count Nodes\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 << "Number of Nodes: " << countNodes(head) << endl;
                break;
        }
    } while (choice != 3);

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

3- Implement a program to delete a node from a doubly linked list based on user input.

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

Delete Node: Deletes a specific node from the doubly linked list. The user will enter the value of the node to be deleted.

Display List: Displays the doubly linked list after each operation.

Exit: Exits the program.

تنفيذ برنامج لحذف عقدة من قائمة مرتبطة بشكل مزدوج بناءً على إدخال المستخدم.

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

حذف العقدة: حذف عقدة معينة من القائمة المرتبطة بشكل مزدوج. سيقوم المستخدم بإدخال قيمة العقدة المراد حذفها.

عرض القائمة: يعرض القائمة المرتبطة بشكل مزدوج بعد كل عملية.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Delete Node
3) Display List
4) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Delete Node
3) Display List
4) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Delete Node
3) Display List
4) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Delete Node
3) Display List
4) Exit
Enter a number: 2
Enter the value of the node to delete: 2
Node deleted with data: 2
Node with value 2 deleted.
1) Add Node
2) Delete Node
3) Display List
4) Exit
Enter a number: 3
Doubly Linked List: 1 3
1) Add Node
2) Delete Node
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* prev;

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

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

Node* addNode(Node* head, int value) {
    Node* newNode = new Node(value);

    if (head == NULL) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
    return head;
}

Node* deleteNode(Node* head, int value) {
    Node* current = head;

    while (current != NULL && current->data != value) {
        current = current->next;
    }

    if (current == NULL) {
        cout << "Node with value " << value << " not found." << endl;
        return head;
    }

    if (current->prev != NULL) {
        current->prev->next = current->next;
    } else {
        head = current->next;
    }

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

    delete current;

    cout << "Node with value " << value << " deleted." << endl;

    return head;
}

void displayList(Node* head) {
    cout << "Doubly 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) Delete Node\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 << "Enter the value of the node to delete: ";
                cin >> value;
                head = deleteNode(head, value);
                break;
            case 3:
                displayList(head);
                break;
        }
    } while (choice != 4);

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

4- Implement a program to check if a linked list is circular or not.

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

Check Circular: Checks if the linked list is circular or not.

Display List: Displays the linked list after each operation.

Exit: Exits the program.

قم بتنفيذ برنامج للتحقق مما إذا كانت القائمة المرتبطة دائرية أم لا.

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

التحقق من التعميم: للتحقق مما إذا كانت القائمة المرتبطة دائرية أم لا.

عرض القائمة: يعرض القائمة المرتبطة بعد كل عملية.

خروج: الخروج من البرنامج.

Input & Output

```
1) Add Node
2) Check Circular
3) Display List
4) Exit
Enter a number: 1
Enter a value: 1
Node created with data: 1
1) Add Node
2) Check Circular
3) Display List
4) Exit
Enter a number: 1
Enter a value: 2
Node created with data: 2
1) Add Node
2) Check Circular
3) Display List
4) Exit
Enter a number: 1
Enter a value: 3
Node created with data: 3
1) Add Node
2) Check Circular
3) Display List
4) Exit
Enter a number: 3
Linked List: 1 2 3
1) Add Node
2) Check Circular
3) Display List
4) Exit
Enter a number: 2
The linked list is not circular.
1) Add Node
2) Check Circular
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) {
    Node* newNode = new Node(value);

    if (head == NULL) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    return head;
}

bool isCircular(Node* head) {
    if (head == NULL) {
        return false;
    }

    Node* slow = head;
    Node* fast = head->next;

    while (fast != NULL && fast->next != NULL) {
        if (slow == fast || slow == fast->next) {
            return true;
        }
        slow = slow->next;
        fast = fast->next->next;
    }

    return false;
}

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) Check Circular\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:
                if (isCircular(head)) {
                    cout << "The linked list is circular." << endl;
                } else {
                    cout << "The linked list is not circular." << endl;
                }
                break;
            case 3:
                displayList(head);
                break;
        }
    } while (choice != 4);

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

5- Write a program to find and print the count of numbers divisible by 3 in the linked list

اكتب برنامجًا للعثور على عدد الأعداد القابلة للقسمة على 3 وطباعته في linked list

Input

```
Enter a number that represents the number of numbers in the
linked list: 5
Enter number: 7
Enter number: 8
Enter number: 9
Enter number: 15
Enter number: 13
```

Output

```
Count of numbers divisible by 3: 2
```

Solution

```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

struct gammal {
    int number;
    struct gammal* next;
};

struct gammal* add(struct gammal* g) {
    if (g == NULL) {
        g = (struct gammal*)malloc(sizeof(struct gammal));
        printf("Enter number: ");
        scanf("%d", &g->number);
        g->next = NULL;
        return g;
    }
    g->next = add(g->next);
    return g;
}

int countDivisibleBy3(struct gammal* g) {
    if (g == NULL) {
        return 0;
    }
    int count = countDivisibleBy3(g->next);
    return (g->number % 3 == 0) ? count + 1 : count;
}

int main() {
    struct gammal* head = NULL;
    int numberSelect;

    printf("Enter a number that represents the number of numbers in the linked list: ");
    scanf("%d", &numberSelect);

    for (int i = 0; i < numberSelect; i++)
        head = add(head);

    int divisibleBy3Count = countDivisibleBy3(head);

    printf("Count of numbers divisible by 3: %d\n", divisibleBy3Count);

    return 0;
}
```

6- Write a program to find and print the sum of all odd numbers in the linked list

اكتب برنامجًا للعثور على مجموع الأعداد الفردية في linked list وطباعته

Input

```
Enter a number that represents the number of numbers in the
linked list: 4
Enter number: 1
Enter number: 2
Enter number: 2
Enter number: 5
```

Output

```
Sum of odd numbers: 6
```

Solution

```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

struct gammal {
    int number;
    struct gammal* next;
};

struct gammal* add(struct gammal* g) {
    if (g == NULL) {
        g = (struct gammal*)malloc(sizeof(struct gammal));
        printf("Enter number: ");
        scanf("%d", &g->number);
        g->next = NULL;
        return g;
    }
    g->next = add(g->next);
    return g;
}

int sumOddNumbers(struct gammal* g) {
    if (g == NULL) {
        return 0;
    }
    int sum = sumOddNumbers(g->next);
    return (g->number % 2 != 0) ? sum + g->number : sum;
}

int main() {
    struct gammal* head = NULL;
    int numberSelect;

    printf("Enter a number that represents the number of numbers in the linked list: ");
    scanf("%d", &numberSelect);

    for (int i = 0; i < numberSelect; i++)
        head = add(head);

    int oddSum = sumOddNumbers(head);

    printf("Sum of odd numbers: %d\n", oddSum);

    return 0;
}
```

7- Write a program to find the minimum element in a double linked list

اكتب برنامجًا للعثور على أقل عنصر في double linked list

Input

```
Enter the number of elements: 5
Enter the elements:
5 1 2 3 4
```

Output

```
Minimum element: 1
```

Solution

```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

struct Node* insertEnd(struct Node* head, int data) {
    if (head == NULL) {
        return createNode(data);
    }
    head->next = insertEnd(head->next, data);
    struct Node* temp = head->next;
    temp->prev = head;
    return head;
}

int findMin(struct Node* head, int min) {
    if (head == NULL) {
        return min;
    }
    min = (head->data < min) ? head->data : min;
    return findMin(head->next, min);
}

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

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    printf("Enter the elements:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        head = insertEnd(head, value);
    }

    int minElement = findMin(head, head->data);
    printf("Minimum element: %d\n", minElement);

    return 0;
}
```

8- Write a program to insert an element at a specific position in a double linked list

اكتب برنامجًا لإدخال عنصر في موضع محدد في double linked list

Input

```
Enter the number of elements: 5
Enter the elements:
2 3 4 5 6
Enter the position to insert at: 3
Enter the new element: 1
```

Output

```
Double Linked List after insertion: 2 <-> 3 <-> 4 <-> 1 <-> 5 <-> 6 <-> NULL
```

Solution

```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

struct Node* insertEnd(struct Node* head, int data) {
    if (head == NULL) {
        return createNode(data);
    }
    head->next = insertEnd(head->next, data);
    struct Node* temp = head->next;
    temp->prev = head;
    return head;
}

struct Node* insertAtPosition(struct Node* head, int position, int data) {
    if (position < 0) {
        printf("Invalid position.\n");
        return head;
    }
    if (position == 0 || head == NULL) {
        struct Node* newNode = createNode(data);
        newNode->next = head;
        if (head != NULL) {
            head->prev = newNode;
        }
        return newNode;
    }
    head->next = insertAtPosition(head->next, position - 1, data);
    if (head->next != NULL) {
        head->next->prev = head;
    }
    return head;
}

void displayList(struct Node* head) {
    if (head == NULL) {
        printf("NULL\n");
        return;
    }
    printf("%d <-> ", head->data);
    displayList(head->next);
}

int main() {
    struct Node* head = NULL;
    int n, value, position, newData;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    printf("Enter the elements:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        head = insertEnd(head, value);
    }

    printf("Enter the position to insert at: ");
    scanf("%d", &position);

    printf("Enter the new element: ");
    scanf("%d", &newData);

    head = insertAtPosition(head, position, newData);

    printf("Double Linked List after insertion: ");
    displayList(head);

    return 0;
}
```

9- Create a circular linked list with user-specified nodes and display the elements and Copy a circular linked list into another circular linked list.

قم بإنشاء circular linked مع العقد المحددة من قبل المستخدم وعرض العناصر ونسخ circular linked إلى circular linked أخرى.

Input & Output

```
1) Add to Original List
2) Copy List
3) Show Original List
4) Show Copied List
Please enter a number: 1
Enter num: 1

1) Add to Original List
2) Copy List
3) Show Original List
4) Show Copied List
Please enter a number: 1
Enter num: 2

1) Add to Original List
2) Copy List
3) Show Original List
4) Show Copied List
Please enter a number: 1
Enter num: 3

1) Add to Original List
2) Copy List
3) Show Original List
4) Show Copied List
Please enter a number: 1
Enter num: 4

1) Add to Original List
2) Copy List
3) Show Original List
4) Show Copied List
Please enter a number: 2
List copied successfully.

1) Add to Original List
2) Copy List
3) Show Original List
4) Show Copied List
Please enter a number: 4
1 2 3 4
```

Solution

```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

struct gammal {
    int num;
    struct gammal* next;
};

struct gammal* add(struct gammal* g, struct gammal* head) {
    if (head == NULL || g->next == head) {
        struct gammal* t = (struct gammal*)malloc(sizeof(struct gammal));
        printf("Enter num: ");
        scanf("%d", &t->num);
        if (head == NULL) {
            t->next = t;
            return t;
        }
        t->next = head;
        g->next = t;
        return g;
    }
    g->next = add(g->next, head);

    return g;
}

struct gammal* copyList(struct gammal* originalHead) {
    if (originalHead == NULL) {
        printf("Original list is empty.\n");
        return NULL;
    }

    struct gammal *originalCurrent = originalHead, *newHead = NULL, *newCurrent = NULL;

    do {
        struct gammal* newNode = (struct gammal*)malloc(sizeof(struct gammal));
        newNode->num = originalCurrent->num;

        if (newHead == NULL) {
            newHead = newNode;
            newCurrent = newNode;
        } else {
            newCurrent->next = newNode;
            newCurrent = newNode;
        }

        originalCurrent = originalCurrent->next;
    } while (originalCurrent != originalHead);

    // Close the circular link
    newCurrent->next = newHead;

    printf("List copied successfully.\n");

    return newHead;
}

void show(struct gammal* g, struct gammal* head) {
    if (g == NULL) {
        printf("List is empty.\n");
        return;
    }

    do {
        printf("%d ", g->num);
        g = g->next;
    } while (g != head);
    printf("\n");
}

int main() {
    int a;
    struct gammal* originalHead = NULL;
    struct gammal* copiedHead = NULL;

    do {
        printf("\n1) Add to Original List\n"
               "2) Copy List\n"
               "3) Show Original List\n"
               "4) Show Copied List\n"
               "Please enter a number: ");
        scanf("%d", &a);

        if (a == 1)
            originalHead = add(originalHead, originalHead);
        else if (a == 2)
            copiedHead = copyList(originalHead);
        else if (a == 3)
            show(originalHead, originalHead);
        else if (a == 4)
            show(copiedHead, copiedHead);
    } while (a);

    return 0;
}
```

10- Write a program that initializes a vector with five integers and finds the sum of all elements.

اكتب برنامجًا يقوم بتهيئة vector بخمسة أعداد صحيحة وإيجاد مجموع جميع العناصر.

Output

```
Sum of vector elements: 15
```

Solution

```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int main() {
    int *vec, i, sum = 0;

    // Initialize vector with five elements
    vec = (int*)malloc(5 * sizeof(int));
    for (i = 0; i < 5; i++)
        vec[i] = i + 1;

    // Calculate the sum of all elements
    for (i = 0; i < 5; i++)
        sum += vec[i];

    // Print the sum
    printf("Sum of vector elements: %d\n", sum);

    // Free allocated memory
    free(vec);

    return 0;
}
```