

```

#include <iostream>

using namespace std;

// Definisi struktur Node
struct Node {
    int data;
    Node* next;
};

// Insert node di awal linked list
void insertAtBeginning(Node*& head, int newData) {
    Node* newNode = new Node;
    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}

// Insert node di akhir linked list
void insertAtEnd(Node*& head, int newData) {
    Node* newNode = new Node;
    newNode->data = newData;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
        return;
    }

    Node* last = head;
    while (last->next != nullptr) {
        last = last->next;
    }
}

```

```

    }

    last->next = newNode;
}

// Hapus node di awal linked list
void deleteAtBeginning(Node*& head) {
    if (head == nullptr) {
        cout << "Linked list kosong. Tidak bisa hapus di awal.\n";
        return;
    }

    Node* temp = head;
    head = head->next;
    delete temp;
}

// Hapus node di akhir linked list
void deleteAtEnd(Node*& head) {
    if (head == nullptr) {
        cout << "Linked list kosong. Tidak bisa hapus di akhir.\n";
        return;
    }

    if (head->next == nullptr) {
        delete head;
        head = nullptr;
        return;
    }

    Node* secondLast = head;
    while (secondLast->next->next != nullptr) {

```

```

        secondLast = secondLast->next;
    }
    delete secondLast->next;
    secondLast->next = nullptr;
}

```

// Cari data tertentu dan ganti dengan nilai baru

```

void updateData(Node* head, int oldData, int newData) {
    Node* current = head;
    while (current != nullptr) {
        if (current->data == oldData) {
            current->data = newData;
            cout << "Data berhasil diupdate.\n";
            return;
        }
        current = current->next;
    }
    cout << "Data tidak ditemukan di linked list.\n";
}

```

// Cetak seluruh isi linked list

```

void printLinkedList(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << ' ';
        current = current->next;
    }
    cout << '\n';
}

```

// Hapus seluruh linked list

```

void deleteLinkedList(Node*& head) {
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
}

```

//Case 7 : Fungsi untuk menghitung jumlah node dalam linked list

```

int countAll(Node* head) {
    int count = 0;
    Node* current = head;
    while (current != nullptr) {
        count++;
        current = current->next;
    }
    return count;
}

```

//Case 8: Fungsi untuk menyisipkan node pada posisi ke-4

```

void insertAs4(Node*& head, int newData) {
    Node* newNode = new Node;
    newNode->data = newData;
    newNode->next = nullptr;
    if (head == nullptr || head->next == nullptr || head->next->next == nullptr) {
        insertAtEnd(head, newData);
        return;
    }
}

```

```

Node* current = head;
int position = 1;

```

```

while (position < 3 && current->next != nullptr) {
    current = current->next;
    position++;
}

newNode->next = current->next;
current->next = newNode;
}

```

// CASE 9: Fungsi untuk mencetak hanya bilangan genap

```

void cetak_genap(Node* head) {
    Node* current = head;
    bool found = false;
    cout << "Data genap dalam list: ";
    while (current != nullptr) {
        if (current->data % 2 == 0) {
            cout << current->data << " ";
            found = true;
        }
        current = current->next;
    }
    if (!found) {
        cout << "(Tidak ada bilangan genap)";
    }
    cout << endl;
}

```

// Case 10: Fungsi untuk menghapus node pada posisi sebelum dua node terakhir

// (kecuali jika elemen linked list < 3)

```

void delbefore2end(Node *&head)

```

```

{
    int total = countAll(head);
    if (total < 3)
    {
        cout << "Elemen linked list kurang dari 3, tidak bisa menghapus." << endl;
        return;
    }

    // Posisi yang dihapus = total - 2 (posisi sebelum 2 node terakhir)
    int posToDelete = total - 2; // 1-indexed

    if (posToDelete == 1)
    {
        // Hapus node pertama (head)
        Node *temp = head;
        head = head->next;
        delete temp;
    }
    else
    {
        // Traverse ke node sebelum posisi yang akan dihapus
        Node *current = head;
        for (int i = 1; i < posToDelete - 1; i++)
        {
            current = current->next;
        }
        Node *temp = current->next;
        current->next = temp->next;
        delete temp;
    }
    cout << "Node pada posisi sebelum 2 node terakhir berhasil dihapus." << endl;
}

```

```
}
```

```
// Tampilkan menu
```

```
void printMenu() {
```

```
    cout
```

```
        << "\nMenu:\n"
```

```
        << "1. Insert di Awal\n"
```

```
        << "2. Insert di Akhir\n"
```

```
        << "3. Hapus di Awal\n"
```

```
        << "4. Hapus di Akhir\n"
```

```
        << "5. Update Data\n"
```

```
        << "6. Cetak Linked List\n"
```

```
        << "7. Hitung Jumlah Node\n"
```

```
        << "8. Insert pada Posisi ke-4\n"
```

```
        << "9. Cetak Bilangan Genap\n"
```

```
        << "10. Hapus Node sebelum 2 Node Terakhir\n"
```

```
        << "0. Keluar\n";
```

```
}
```

```
int main() {
```

```
    Node* head = nullptr;
```

```
    int choice;
```

```
    do {
```

```
        printMenu();
```

```
        cout << "Pilih operasi (0-10): ";
```

```
        cin >> choice;
```

```
        switch (choice) {
```

```
            case 1: {
```

```
                int insertData;
```

```

    cout << "Masukkan data untuk di-insert di awal: ";

    cin >> insertData;

    insertAtBeginning(head, insertData);

    break;
}

case 2: {

    int insertDataEnd;

    cout << "Masukkan data untuk di-insert di akhir: ";

    cin >> insertDataEnd;

    insertAtEnd(head, insertDataEnd);

    break;
}

case 3: {

    deleteAtBeginning(head);

    break;
}

case 4: {

    deleteAtEnd(head);

    break;
}

case 5: {

    int oldData, newData;

    cout << "Masukkan data yang ingin di-update: ";

    cin >> oldData;

    cout << "Masukkan nilai baru: ";

    cin >> newData;

    updateData(head, oldData, newData);

    break;
}

case 6: {

    printLinkedList(head);

```

```

        break;
    }
    case 7:
    {
        int total = countAll(head);
        cout << "Jumlah node dalam linked list: " << total << endl;
        break;
    }
    case 8:
    {
        int newData;
        cout << "Masukkan data untuk di-insert pada posisi ke-4: ";
        cin >> newData;
        insertAs4(head, newData);
        break;
    }
    case 9:
    {
        cetak_genap(head);
        break;
    }
    case 10:
    {
        delbefore2end(head);
        break;
    }
    case 0: {
        cout << "Keluar dari program.\n";
        break;
    }
    default: {

```

```
        cout << "Pilihan tidak valid. Coba lagi.\n";  
        break;  
    }  
}  
} while (choice != 0);  
  
deleteLinkedList(head);  
return 0;  
}
```