Data Structures with C++ Practice Log

//Linked List: Inserting a Node at the Beginning

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

Node\* head;

void Insert(int x){

    Node\* temp = new Node();

    temp->data = x;

    temp->next = head;

    head = temp;

}

void Print(){

    Node\* temp = head;

    std::cout << "The list is : " << std::endl;

    while(temp != nullptr) {

        std::cout << temp->data << " ";

        temp = temp->next;

    }

    std::cout << std::endl;

}

int main() {

    int n, x;

    head = nullptr;

    std::cout << "How many numbers?" << std::endl;

    std::cin >> n;

    for (int i = 0; i < n; i++){

        std::cout << "Enter the number" << std::endl;

        std::cin >> x;

        Insert(x);

        Print();

    }

    return 0;

}

//Linked List Insertion at Beginning With Local Head

#include <iostream>

    struct Node {

        int data;

        Node\* next;

    };

    Node\* insert(Node\* head, int x){

        Node\* temp = new Node();

        temp -> data = x;

        temp -> next = head;

        head = temp;

        return head;

    }

    void print(Node\* head) {

        std::cout << "The current list is : ";

        while (head != nullptr) {

            std::cout << head->data << " ";

            head = head->next;

        }

        std::cout << std::endl;

    }

int main(){

    Node\* head = nullptr;

    int n, x;

    std::cout << "How many numbers would you like to include?" << std::endl;

    std::cin >> n;

    for (int i{0}; i < n; i++){

        std::cout << "What number would you like to input?" << std::endl;

        std::cin >> x;

        head = insert(head, x);

        print(head);

    }

    return 0;

}

//Linked List insertion at the N-th Position

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

Node\* head;

void insert(int data, int n){

    Node\* temp1 = new Node();

    temp1 -> data = data;

    temp1 -> next = nullptr;

    if (n==1) {

        temp1 -> next = head;

        head = temp1;

        return;

    }

    Node\* temp2 = head;

    for (int i {0}; i < n-2; i++){

        temp2 = temp2 -> next;

    }

    temp1 -> next = temp2 -> next;

    temp2 -> next = temp1;

}

void print(){

    Node\* temp = head;

    while (temp != nullptr){

        std::cout << temp->data << " ";

        temp = temp -> next;

    }

    std::cout<<std::endl;

}

int main(){

    head = nullptr; //empty list

    insert(2,1); //list: 2

    insert(3,2); //list: 2, 3

    insert(4,1); //list: 4, 2, 3

    insert(5,2); //list: 4, 5, 2, 3

    print();

    return 0;

}

//Linked List : Delete a node at the nth position and Insert at the end

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

Node\* head;

void insert(int data) {

    Node\* temp1 = new Node();

    temp1 -> data = data;

    if (head == nullptr) {

        head = temp1;

        return;

    }

    Node\* temp2 = head;

    while (temp2 -> next != nullptr) {

        temp2 = temp2 -> next;

    }

    temp2 -> next= temp1;

}

void print() {

    Node\* temp1 = head;

    while (temp1 != nullptr) {

        std::cout << temp1 -> data << " ";

        temp1 = temp1 -> next;

    }

    std::cout << std::endl;

}

void delete\_node(int n) {

    Node\* temp1 = head;

    if(n==1) {

        head = temp1->next;

        delete temp1;

        temp1 = nullptr;

        return;

    }

    for(size\_t i {}; i < n-2; i++) {

        temp1 = temp1 -> next;

    }

    Node\* temp2 = temp1 -> next;

    temp1 -> next = temp2 -> next;

    delete temp2;

    temp2 = nullptr;

}

int main() {

    head = nullptr;

    int n;

    insert(2);

    insert(4);

    insert(6);

    insert(5);

    print();

    std::cout << "Enter a position" << std::endl;

    std::cin >> n;

    delete\_node(n);

    print();

    return 0;

}

//reverse linked list - iterative method

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

Node\* head;

void reverse() {

    Node\* current = head;

    Node\* prev = nullptr;

    Node\* next = nullptr;

    while (current != nullptr) {

        next = current -> next;

        current -> next = prev;

        prev = current;

        current = next;

    }

    head = prev;

}

void insert(int data){

    Node\* temp1 = new Node();

    temp1 -> data = data;

    if (head == nullptr) {

        temp1 -> next = head;

        head = temp1;

        return;

    }

    Node\* temp2 = head;

    while (temp2 -> next != nullptr) {

        temp2 = temp2 -> next;

    }

    temp2 -> next = temp1;

}

void print() {

    Node\* temp1 = head;

    while (temp1 != nullptr) {

        std::cout << temp1->data << " ";

        temp1 = temp1 -> next;

    }

    std::cout << std::endl;

}

int main () {

    head = nullptr;

    insert(5);

    insert(1);

    insert(8);

    insert(3);

    insert(3);

    insert(6);

    insert(4);

    print();

    reverse();

    print();

}

//linked list: // reverse the linked list // print using recursion forwards and backwards

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

Node\* head;

void insert(int data) {

    Node\* temp1 = new Node();

    temp1 -> data = data;

    temp1 -> next = nullptr;

    if (head == nullptr) {

        temp1 -> next = head;

        head = temp1;

        return;

    }

    Node\* temp2 = head;

    while (temp2 -> next != nullptr) {

        temp2 = temp2 -> next;

    }

    temp2 -> next= temp1;

}

void print(Node\* head) {

    Node\* temp1 = head;

    if (temp1 == nullptr) return;

    std::cout << temp1 -> data << " ";

    print(temp1->next);

}

void reverse\_list(){

    Node\* curr = head;

    Node\* prev = nullptr;

    Node\* next = nullptr;

    while (curr != nullptr) {

        next = curr -> next;

        curr -> next = prev;

        prev = curr;

        curr = next;

    }

    head = prev;

}

void reverse\_print(Node\* head) {

    Node\* temp1 = head;

    if (temp1 == nullptr) return;

    reverse\_print(temp1 -> next);

    std::cout << temp1 -> data << " ";

}

int main() {

    head = nullptr;

    int n;

    insert(5);

    insert(2);

    insert(9);

    insert(5);

    insert(2);

    print(head);

    std::cout << std::endl;

    std::cout << "Reversed List : " ;

    // reverse\_list();

    // print(head);

    reverse\_print(head);

    return 0;

}