DSA ASSIGNMENT 5

1.)

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class SinglyLinkedList {

Node\* head;

public:

SinglyLinkedList() { head = nullptr; }

void insertAtBeginning(int val) {

Node\* temp = new Node{val, head};

head = temp;

}

void insertAtEnd(int val) {

Node\* temp = new Node{val, nullptr};

if (!head) { head = temp; return; }

Node\* t = head;

while (t->next) t = t->next;

t->next = temp;

}

void insertBefore(int key, int val) {

if (!head) return;

if (head->data == key) { insertAtBeginning(val); return; }

Node\* t = head;

while (t->next && t->next->data != key) t = t->next;

if (t->next) {

Node\* temp = new Node{val, t->next};

t->next = temp;

}

}

void insertAfter(int key, int val) {

Node\* t = head;

while (t && t->data != key) t = t->next;

if (t) {

Node\* temp = new Node{val, t->next};

t->next = temp;

}

}

void deleteBeginning() {

if (!head) return;

Node\* temp = head;

head = head->next;

delete temp;

}

void deleteEnd() {

if (!head) return;

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

Node\* t = head;

while (t->next->next) t = t->next;

delete t->next;

t->next = nullptr;

}

void deleteNode(int key) {

if (!head) return;

if (head->data == key) { deleteBeginning(); return; }

Node\* t = head;

while (t->next && t->next->data != key) t = t->next;

if (t->next) {

Node\* del = t->next;

t->next = t->next->next;

delete del;

}

}

void search(int key) {

int pos = 1;

Node\* t = head;

while (t) {

if (t->data == key) { cout << "Found at position " << pos << endl; return; }

pos++;

t = t->next;

}

cout << "Not Found\n";

}

void display() {

Node\* t = head;

while (t) { cout << t->data << " "; t = t->next; }

cout << endl;

}

};

int main() {

SinglyLinkedList s;

int ch, val, key;

do {

cout << "\n1.InsertBeg 2.InsertEnd 3.InsertBefore 4.InsertAfter 5.DeleteBeg 6.DeleteEnd 7.DeleteNode 8.Search 9.Display 10.Exit\n";

cin >> ch;

switch(ch) {

case 1: cin >> val; s.insertAtBeginning(val); break;

case 2: cin >> val; s.insertAtEnd(val); break;

case 3: cin >> key >> val; s.insertBefore(key,val); break;

case 4: cin >> key >> val; s.insertAfter(key,val); break;

case 5: s.deleteBeginning(); break;

case 6: s.deleteEnd(); break;

case 7: cin >> key; s.deleteNode(key); break;

case 8: cin >> key; s.search(key); break;

case 9: s.display(); break;

}

} while(ch!=10);

return 0;

}

2.)

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void insert(Node\*& head, int val) {

Node\* temp = new Node{val, nullptr};

if (!head) { head = temp; return; }

Node\* t = head;

while (t->next) t = t->next;

t->next = temp;

}

void countAndDelete(Node\*& head, int key) {

int count = 0;

while (head && head->data == key) {

Node\* temp = head;

head = head->next;

delete temp;

count++;

}

Node\* t = head;

while (t && t->next) {

if (t->next->data == key) {

Node\* del = t->next;

t->next = t->next->next;

delete del;

count++;

} else t = t->next;

}

cout << "Count: " << count << endl;

}

void display(Node\* head) {

while (head) { cout << head->data; if (head->next) cout << "->"; head = head->next; }

cout << endl;

}

int main() {

Node\* head = nullptr;

int n,val,key;

cin >> n;

for(int i=0;i<n;i++){ cin>>val; insert(head,val); }

cin >> key;

countAndDelete(head,key);

display(head);

return 0;

}

3.)

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void insert(Node\*& head, int val) {

Node\* temp = new Node{val, nullptr};

if (!head) { head = temp; return; }

Node\* t = head;

while (t->next) t = t->next;

t->next = temp;

}

int findMiddle(Node\* head) {

Node\* slow = head;

Node\* fast = head;

while (fast && fast->next) {

slow = slow->next;

fast = fast->next->next;

}

return slow ? slow->data : -1;

}

int main() {

Node\* head = nullptr;

int n,val;

cin >> n;

for(int i=0;i<n;i++){ cin>>val; insert(head,val); }

cout << findMiddle(head) << endl;

return 0;

}

4.)

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void insert(Node\*& head, int val) {

Node\* temp = new Node{val, nullptr};

if (!head) { head = temp; return; }

Node\* t = head;

while (t->next) t = t->next;

t->next = temp;

}

void reverse(Node\*& head) {

Node\* prev = nullptr;

Node\* curr = head;

while (curr) {

Node\* next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

head = prev;

}

void display(Node\* head) {

while (head) { cout << head->data; if (head->next) cout << "->"; head = head->next; }

cout << "->NULL" << endl;

}

int main() {

Node\* head = nullptr;

int n,val;

cin >> n;

for(int i=0;i<n;i++){ cin>>val; insert(head,val); }

reverse(head);

display(head);

return 0;

}