Bai 1: Test:

7

1

CNTT1

Le Van Long

2.40

1

CNTT2

Le Van Long

1.20

1

CNTT3

Vuong Hoang Manh

1.20

1

CNTT4

Le Hoang Nam

2.80

2

CNTT5

Vuong Van Phuoc

2.60

1

CNTT6

Pham Xuan Manh

3.10

3 2

CNTT7

Le Xuan Phuoc

2.40

Bai 2: Test

8 7

6 7 1 9 4 5 4 7

Hàm xóa toàn bộ node có data biết trước:

void xoa (node\*&head, int x) {

    if (head == nullptr) return;

    while (head != NULL && head->data == x) {

        node\* tmp = head;

        head = head->next;

**delete** tmp;

    }

    node\* prev = head;

    node\* back = head;

    while (prev != NULL) {

        if (prev->data == x) {

            back->next = prev->next;

**delete** prev;

            prev = back->next;

        }

        else {

            back = prev;

            prev = prev->next;

        }

    }

}

Code bai 8

1x^0 + 4x^3 + 5x^2 + 5x^1 + 1x^0 + 1x^1 + 3x^2 + 3x^3 + 1x^1

2x^0 + 5x^4 + 4x^0 + 4x^1 + 1x^0 + 2x^0 + 1x^3 + 3x^2 + 3x^3

#include <bits/stdc++.h>

using namespace std;

struct node {

    int heso, somu;

    node\* next;

};

node\* makeNode (int heso, int somu) {

    node\* newNode = **new** node;

    newNode->heso = heso;

    newNode->somu = somu;

    newNode->next = NULL;

    return newNode;

}

void duyet (node\* head) {

    while (head != NULL) {

        cout << head->heso << "x^" << head->somu;

        head = head->next;

        if (head != NULL) {

            cout << " + ";

        }

    }

}

void them(node\*& head, int heso, int somu) {

    node\* truoc = head;

    node\* sau = head;

    while (truoc != NULL) {

        if (truoc->somu == somu) {

            truoc->heso += heso;

            return;

        }

        sau = truoc;

        truoc = truoc->next;

    }

    node\* newNode = makeNode(heso, somu);

    if (sau == NULL) {

        head = newNode;

        return;

    }

    sau->next = newNode;

}

void sapxep(node\*& head) {

    for (node \*i = head; i != NULL; i = i->next) {

        node\* max = i;

        for (node\*j = i->next; j != NULL; j = j->next) {

            if (max->somu < j->somu) {

                max = j;

            }

        }

        swap(i->heso, max->heso);

        swap(i->somu, max->somu);

    }

}

int main () {

    string s;

    node\* head = NULL;

    while (cin >> s) {

        if (s.size() > 1) {

            int heso = (s[0] - '0');

            int somu = (s[3] - '0');

            them(head, heso, somu);

        }

    }

    sapxep(head);

    duyet(head);

}

Cú pháp nhập mảng ký tự

Cin.getline(tên mảng, số ký tự tối đa)

#include <bits/stdc++.h>

using namespace std;

struct node {

    char \*id;

    char \* name;

    double gpa;

    node\* next;

};

double max\_value = INT\_MIN;

node \* makeNode () {

    node\* newnode = **new** node;

    newnode->id = **new** char [100];

    newnode->name = **new** char[100];

    newnode->next = NULL;

    cin.ignore();

    cin.getline(newnode->id, 100);

    cin.getline(newnode->name, 100);

    cin >> newnode->gpa;

    max\_value = max (max\_value, newnode->gpa);

    return newnode;

}

void them(node\*&head) {

    node\* newnode = makeNode();

    if (head == NULL) {

        head = newnode;

        return;

    }

    node\* tmp = head;

    while (tmp->next != NULL) {

        tmp = tmp->next;

    }

    tmp->next = newnode;

}

int main () {

    node\* head = NULL;

    int n; cin >> n;

    while (n--) {

        them(head);

    }

    while (head != NULL) {

        if (head->gpa == max\_value) {

            cout << head->id << " " << head->name << ' ' << head->gpa << endl;

        }

        head = head->next;

    }

}

#include <bits/stdc++.h>

using namespace std;

struct node {

    int data;

    node\* next;

};

node\* makeNode (int x){

    node\* newNode = **new** node;

    newNode->data = x;

    newNode->next = NULL;

    return newNode;

}

void themdau(node\*&head, int x){

    node\* newNode = makeNode(x);

    newNode->next = head;

    head = newNode;

}

void themcuoi(node\*&head, int x){

    node\* newNode = makeNode(x);

    if (head == NULL){

        head = newNode;

        return;

    }

    node\* tmp = head;

    while(tmp->next == NULL){

        tmp = tmp->next;

    }

    tmp->next = newNode;

}

int countNode (node\*&head){

    int n = 0;

    while (head != NULL){

        n++;

        head = head->next;

    }

    return n;

}

void themgiua (node\*&head, int x, int k){

    int n = countNode(head);

    if (n < 1 || n > k + 1) return;

    if (n == 1) themdau(head, x);

    node\*newNode = makeNode(x);

    node\* tmp = head;

    for (int i = 1; i <= k - 2; i++){

        tmp = tmp->next;

    }

    newNode->next = tmp->next;

    tmp->next = newNode;

}

void xoadau(node\*&head){

    if (head == NULL){

        return;

    }

    node\* tmp = head;

    head = tmp->next;

**delete**  tmp;

}

void xoacuoi(node\*&head){

    if (head == NULL) return;

    node\* tmp = head;

    if (head->next == NULL) {

        head = NULL;

**delete** tmp;

        return;

    }

    while (tmp->next->next != NULL){

        tmp = tmp->next;

    }

    node\* lastNode = tmp->next;

    tmp->next = NULL;

**delete** lastNode;

}

void xoagiua(node\*&head, int k){

    int n = countNode(head);

    if (k < 1 || k > n) return;

    if (k == 1){

        xoadau(head);

        return;

    }

    node \*prev = head;

    node\* back = head;

    for (int i = 1; i <= k - 1; i++){

        back = prev;

        prev = prev->next;

    }

    back->next = prev->next;

**delete** prev;

}

void sort (node\*&head){

    // selection

    for (node\*i = head; i != NULL; i = i->next){

        node\* min = i;

        for (node\* j = i; j != NULL; j = j->next){

            if (j->data > min->data){

                min = j;

            }

        }

        swap(i->data, min->data);

    }

}

int main(){

    node\* head = NULL;

}

Double list

#include <bits/stdc++.h>

using namespace std;

struct node {

    int data;

    node\* prev;

    node\* next;

};

node\* makeNode (int x) {

    node\* newNode = **new** node;

    newNode->data = x;

    newNode->prev = NULL;

    newNode->next = NULL;

    return newNode;

}

int countNode(node\*head){

    int n = 0;

    while(head != NULL){

        head = head->next;

        n++;

    }

    return n;

}

void duyetthuan(node\*head){

    while (head != NULL){

        cout << head->data << ' ';

        head = head->next;

    }

}

void duyetnguoc(node\*head){

    while (head != NULL){

        head = head->next;

    }

    node\* tmp = head->prev;

    while( tmp!= NULL ){

        cout << tmp->data << ' ';

        tmp = tmp->prev;

    }

}

void themdau(node\*&head, int x){

    node\* newNode = makeNode(x);

    if (head == NULL){

        head = newNode;

        return;

    }

    newNode->next = head;

    head->prev = newNode;

    head = newNode;

}

void themcuoi (node\*&head, int x){

    node\* newNode = makeNode(x);

    if (head == NULL){

        head = newNode;

        return;

    }

    node\* tmp = head;

    while(tmp->next != NULL){

        tmp = tmp->next;

    }

    tmp->next = newNode;

    newNode->prev = tmp;

}

void themgiua(node\*&head, int x, int k){

    int n = countNode(head);

    if (k < 1 || k > n + 1){

        return;

    }

    if (k == 1){

        themdau(head, x);

        return;

    }

    node\*newNode = makeNode(x);

    node\* tmp = head;

    for (int i = 1; i <= k - 2; i++){

        tmp = tmp->next;

    }

    newNode->next = tmp->next;

    tmp->next->prev = newNode;

    tmp->next = newNode;

    newNode->prev = tmp;

}

void xoadau(node\*&head){

    if (head == NULL){

        return;

    }

    node\* tmp = head;

    head = tmp->next;

    head->prev = NULL;

    delete tmp;

}

void xoacuoi(node\*&head){

    if (head == NULL){

        return;

    }

    node\* tmp = head;

    while (tmp->next->next != NULL){

        tmp = tmp->next;

    }

    node\* lastNode = tmp->next;

    tmp->next = NULL;

    delete lastNode;

}

void xoagiua(node\*&head, int k){

    int n = countNode(head);

    if (k < 1 || k > n) return;

    if (k == 1) xoadau(head); return;

    node\*truoc = head;

    node\*sau = head;

    for (int i = 1; i <= k - 1; i++){

        sau = truoc;

        truoc = truoc->next;

    }

    sau->next = truoc->next;

    truoc->next->prev = sau;

    delete truoc;

}

int main(){

}