# Bài thực hành 4: Thực hành sử dụng các cấu trúc dữ liệu cơ bản để giải quyết các bài toán cụ thể

Contents

[Bài thực hành 4: Thực hành sử dụng các cấu trúc dữ liệu cơ bản để giải quyết các bài toán cụ thể 1](#_Toc72674844)

[Phần 1: Bài tập thực hành 3](#_Toc72674845)

[Bài tập 1: Đảo ngược một danh sách liên kết đơn 3](#_Toc72674846)

[Hãy hoàn thiện các hàm thao tác trên một danh sách liên kết: 3](#_Toc72674847)

[Thêm một phần tử vào đầu danh sách liên kết 3](#_Toc72674848)

[In danh sách 3](#_Toc72674849)

[Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian O(N) và chi phí bộ nhớ dùng thêm O(1)) 3](#_Toc72674850)

[Bài tập 2: Tính diện tích tam giác 5](#_Toc72674851)

[Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh. Trong đó, Point là kiểu được định nghĩa sẵn trong trình chấm như sau: using Point = pair<double, double>; 5](#_Toc72674852)

[Bài tập 3: Tính tích có hướng của 2 vector 6](#_Toc72674853)

[Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product(Vector a, Vector b) tính tích có hướng của 2 vector. Trong đó Vector là kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau: using Vector = tuple<double, double, double>; 6](#_Toc72674854)

[Bài tập 4: Thao tác với vector 7](#_Toc72674855)

[Cho hai vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần. 7](#_Toc72674856)

[Bài tập 5: 10](#_Toc72674857)

[Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > . Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra). 10](#_Toc72674858)

[Bài tập 6: 12](#_Toc72674859)

[Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > . Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra). 12](#_Toc72674860)

[Bài tập 7: 14](#_Toc72674861)

[Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set. 14](#_Toc72674862)

[Bài tập 8: 16](#_Toc72674863)

[Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. 16](#_Toc72674864)

[Trong đó mỗi phần tử được gán cho một số thực trong đoạn [0..1] biểu thị độ thuộc của phần tử trong tập hợp, với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 0 nghĩa là phần tử chắc chắn không thuộc trong tập hợp. 16](#_Toc72674865)

[Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp, với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc. 16](#_Toc72674866)

[Bài tập 9: 19](#_Toc72674867)

[Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng priority\_queue Cụ thể, bạn cần cài đặt hàm vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) nhận đầu vào là danh sách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng số tương ứng của cạnh. Đồ thị gồm n đỉnh được đánh số từ 0 tới n-1. Hàm cần trả vector<int> chứa n phần tử lần lượt là khoảng cách đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ..., n-1. 19](#_Toc72674868)

[Phần 2: Bài tập về nhà 21](#_Toc72674869)

[Bài tập 10: Search Engine 21](#_Toc72674870)

[Xây dựng một máy tìm kiếm (search engine) đơn giản. 21](#_Toc72674871)

[Cho NN văn bản và QQ truy vấn. Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất. 21](#_Toc72674872)

[Sử dụng phương pháp tính điểm TF-IDF 21](#_Toc72674873)

[Bài tập 11. Bảo vệ lâu đài 26](#_Toc72674874)

[Bức tường bao quanh một lâu đài nọ được cấu thành từ nn đoạn tường được đánh số từ 1 đến nn. Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi aiai tên giặc đánh vào đoạn tường thứ ii. Để bảo vệ lâu đài có tất cả ss lính. 26](#_Toc72674875)

[Do các đoạn tường có chất lượng khác nhau nên khả năng bảo vệ tại các đoạn tường cũng khác nhau. Cụ thể tại đoạn tường thứ ii, mỗi lính có thể đẩy lùi tấn công của kiki tên giặc. 26](#_Toc72674876)

[Giả sử đoạn tường thứ ii có xixi lính. Khi đó nếu số tên giặc không vượt quá xi×kixi×ki thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai−xi×kiai−xi×ki tên giặc lọt vào lâu đài qua đoạn tường này. 26](#_Toc72674877)

[Yêu cầu hãy viết chương trình phân bố lính đứng ở các đoạn tường sao cho tổng số lính là ss và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất. 26](#_Toc72674878)

[Bài tập 12. Lược đồ 28](#_Toc72674879)

[Cho một lược đồ gồm nn cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hnh1,h2,…,hn. Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp. 28](#_Toc72674880)

[Bài tập 13: Đếm xâu con 29](#_Toc72674881)

[Cho một xâu nhị phân độ dài nn. Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau. 29](#_Toc72674882)

## Phần 1: Bài tập thực hành

### Bài tập 1: Đảo ngược một danh sách liên kết đơn

### Hãy hoàn thiện các hàm thao tác trên một danh sách liên kết:

### Thêm một phần tử vào đầu danh sách liên kết

### In danh sách

### Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian O(N) và chi phí bộ nhớ dùng thêm O(1))

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap1\_Buoi4.cpp

#include <iostream>

using namespace std;

struct *Node* {

    int data;

*Node*\* next;

    Node(int *data*) {

*this*->data = *data*;

        next = NULL;

    }

};

// push a new element to the beginning of the list

*Node*\* prepend(*Node*\* *head*, int *data*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*Node* \*new\_node = **new** *Node*(*data*);

    new\_node->next = *head*;

*head* = new\_node;

    return *head*;

}

// print the list content on a line

void print(*Node*\* *head*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*Node* \*tmp = *head*;

    while (tmp != NULL) {

        cout << tmp->data << " ";

        tmp = tmp->next;

    }

}

// return the new head of the reversed list

*Node*\* reverse(*Node*\* *head*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*Node* \*cur = *head*, \*pre = NULL; // pre tro toi node lien truoc cua node tro boi cur

*Node* \*tmp = NULL;

    while (cur != NULL) {

        tmp = cur->next;

        cur->next = pre;

        pre = cur;

        cur = tmp;

    }

*head* = pre;

    return *head*;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    int n, u;

    cin >> n;

*Node*\* head = NULL;

    for (int i = 0; i < n; ++i){

        cin >> u;

        head = prepend(head, u);

    }

    cout << "Original list: ";

    print(head);

    cout << endl;

    head = reverse(head);

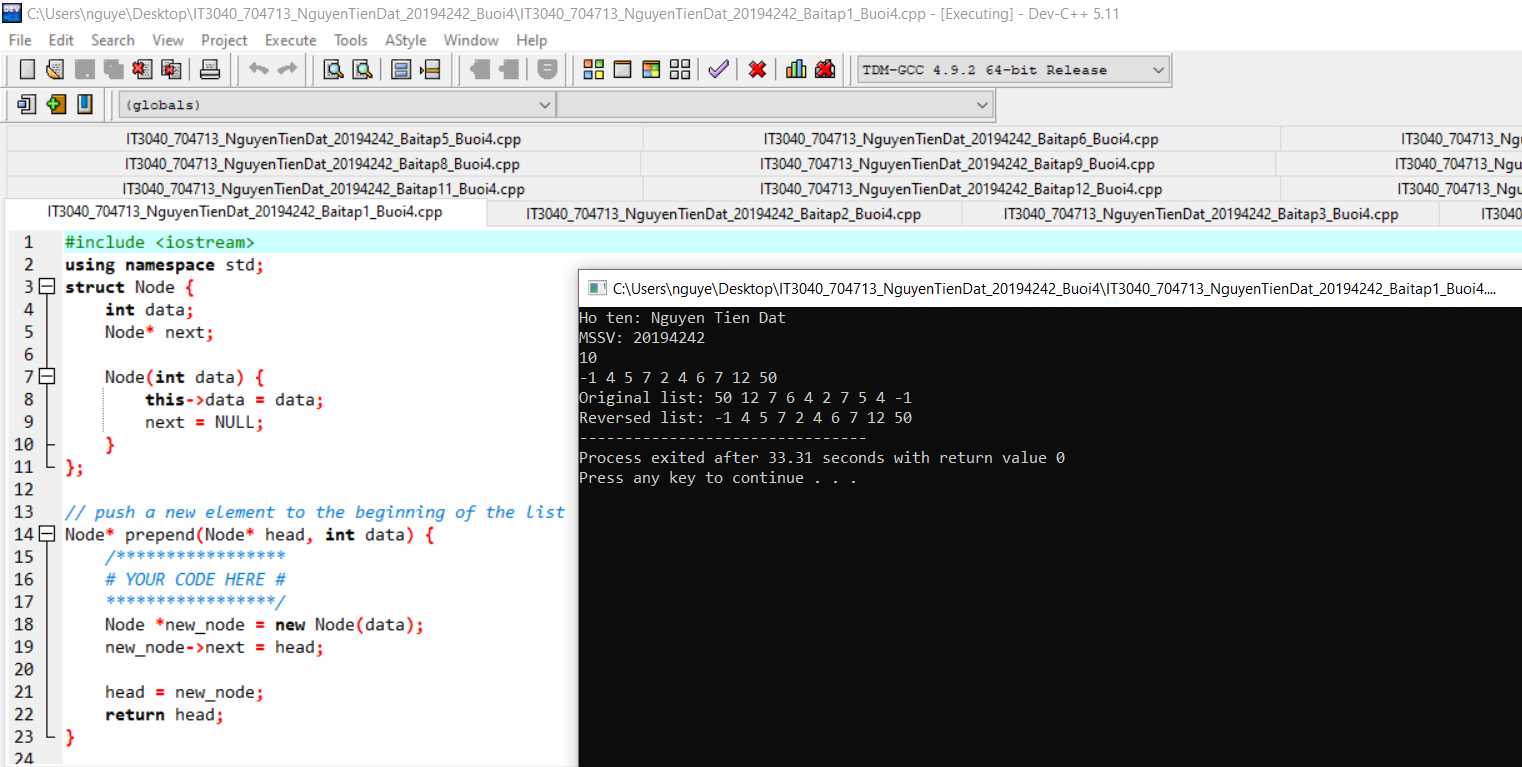
    cout << "Reversed list: ";

    print(head);

    return 0;

}

Kết quả:



### Bài tập 2: Tính diện tích tam giác

### Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh. Trong đó, Point là kiểu được định nghĩa sẵn trong trình chấm như sau: using Point = pair<double, double>;

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap2\_Buoi4.cpp

 #include <iostream>

 #include <cmath>

 #include <iomanip>

 #include <utility>

 using namespace std;

 using *Point* = pair<double, double>;

double area(*Point* *a*, *Point* *b*, *Point* *c*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    return 0.5 \* fabs((*c*.first - *a*.first) \* (*b*.second - *a*.second) - (*b*.first - *a*.first) \* (*c*.second - *a*.second));

}

 int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

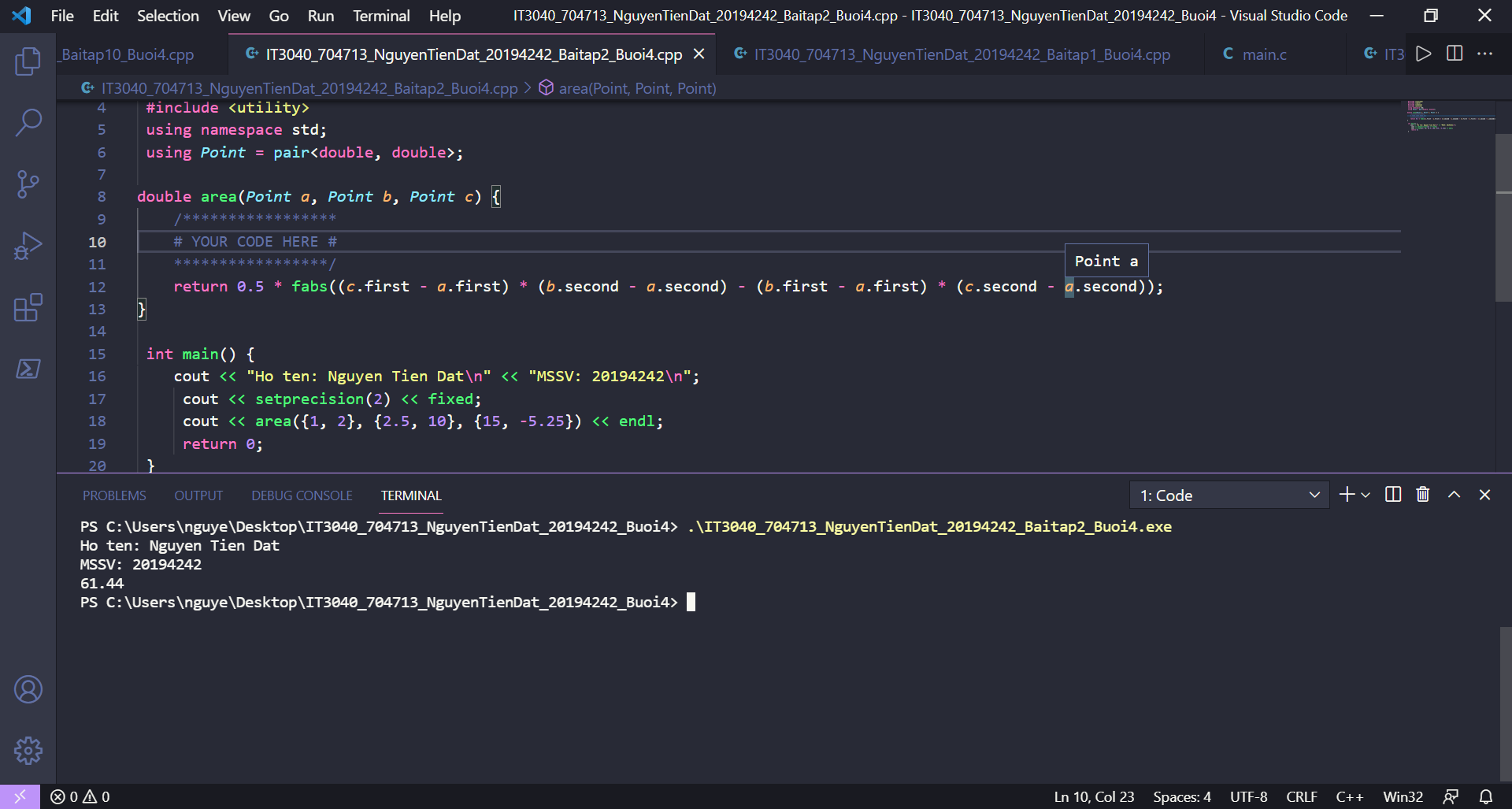
     cout << setprecision(2) << fixed;

     cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;

     return 0;

 }

Kết quả:



### Bài tập 3: Tính tích có hướng của 2 vector

### Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product(Vector a, Vector b) tính tích có hướng của 2 vector. Trong đó Vector là kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau: using Vector = tuple<double, double, double>;

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap3\_Buoi4.cpp

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

using *Vector* = tuple<double, double, double>;

*Vector* cross\_product(*Vector* *a*, *Vector* *b*) {

   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   # YOUR CODE HERE #

   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   double x = get<1>(*a*) \* get<2>(*b*) - get<2>(*a*) \* get<1>(*b*);

    double y = get<2>(*a*) \* get<0>(*b*) - get<0>(*a*) \* get<2>(*b*);

    double z = get<0>(*a*) \* get<1>(*b*) - get<1>(*a*) \* get<0>(*b*);

    return {x, y, z};

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    cout << setprecision(2) << fixed;

*Vector* a {1.2, 4, -0.5};

*Vector* b {1.5, -2, 2.5};

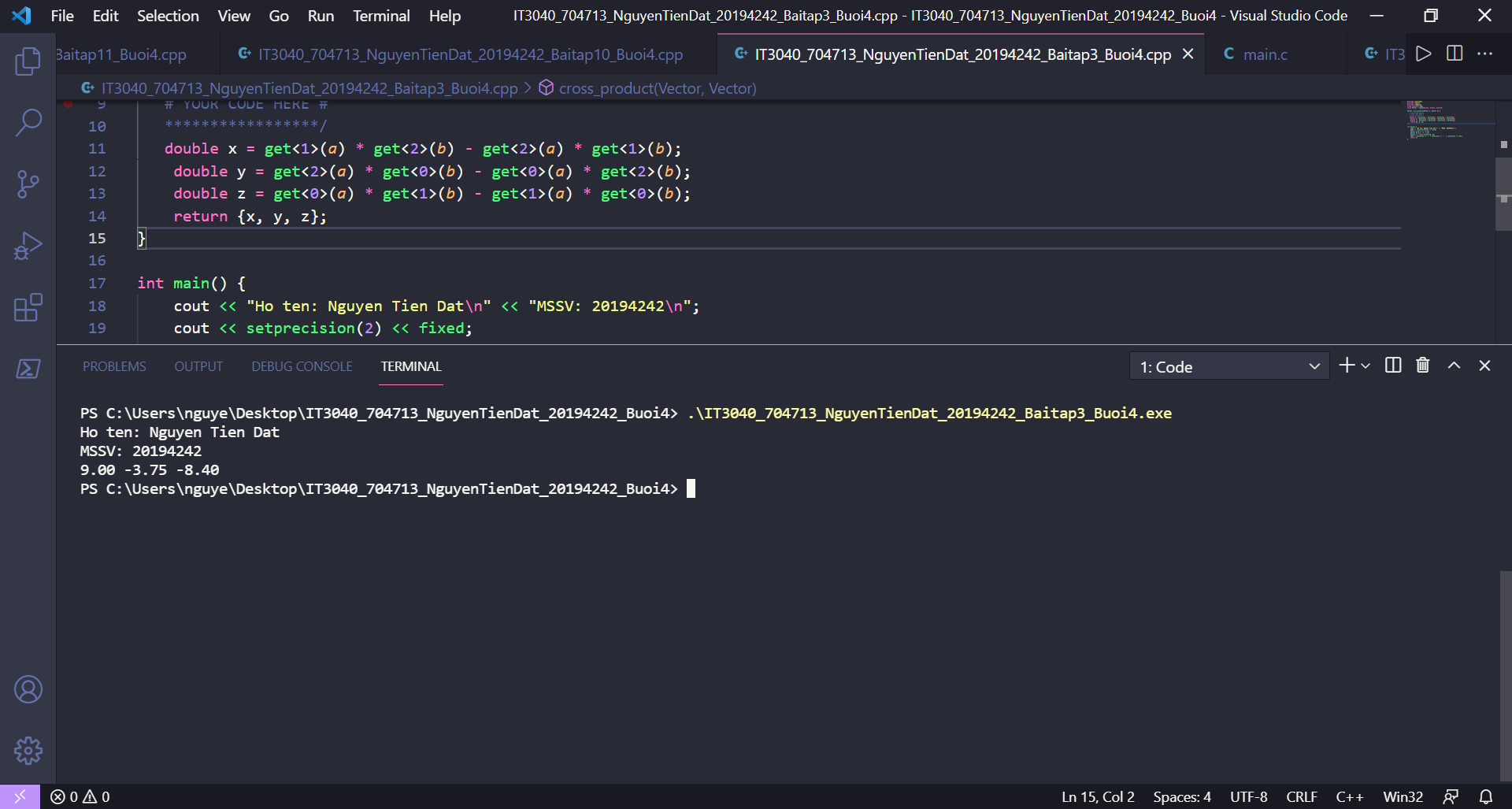
*Vector* c = cross\_product(a, b);

    cout << get<0>(c) << ' ' << get<1>(c) << ' ' << get<2>(c) << endl;

    return 0;

}

Kết quả:



### Bài tập 4: Thao tác với vector

### Cho hai vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap4\_Buoi4.cpp

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void print\_vector(const vector<int> &*a*) {

    for (int v : *a*) cout << v << ' ';

    cout << endl;

}

bool isEven(int *i*) { return ((*i*%2) == 0); }

void delete\_even(vector<int> &*a*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    for (vector<int>::*iterator* i = *a*.begin(); i != *a*.end(); ++i) {

        if (\*i % 2 == 0) {

*a*.erase(i);

            i--;

        }

    }

}

bool isDecrease(int *l*, int *r*) { return *l* > *r*;}

void sort\_decrease(vector<int> &*a*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   sort(*a*.begin(), *a*.end(), isDecrease);

}

vector<int> merge\_vectors(const vector<int> &*a*, const vector<int> &*b*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   vector<int> c;

   // merge(a.begin(), a.end(), b.begin(), b.end(), c.begin(), greaters());

    for(int v : *a*){

        c.push\_back(v);

    }

    for(int v : *b*){

        c.push\_back(v);

    }

    sort(c.begin(), c.end(), isDecrease);

   return c;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    int m, n, u;

    std::vector<int> a, b;

    std::cin >> m >> n;

    for(int i = 0; i < m; i++){

        std:: cin >> u;

        a.push\_back(u);

    }

    for(int i = 0; i < n; i++){

        std:: cin >> u;

        b.push\_back(u);

    }

    delete\_even(a);

    cout << "Odd elements of a: ";

    print\_vector(a);

    delete\_even(b);

    cout << "Odd elements of b: ";

    print\_vector(b);

    sort\_decrease(a);

    cout << "Decreasingly sorted a: ";

    print\_vector(a);

    sort\_decrease(b);

    cout << "Decreasingly sorted b: ";

    print\_vector(b);

    vector<int> c = merge\_vectors(a, b);

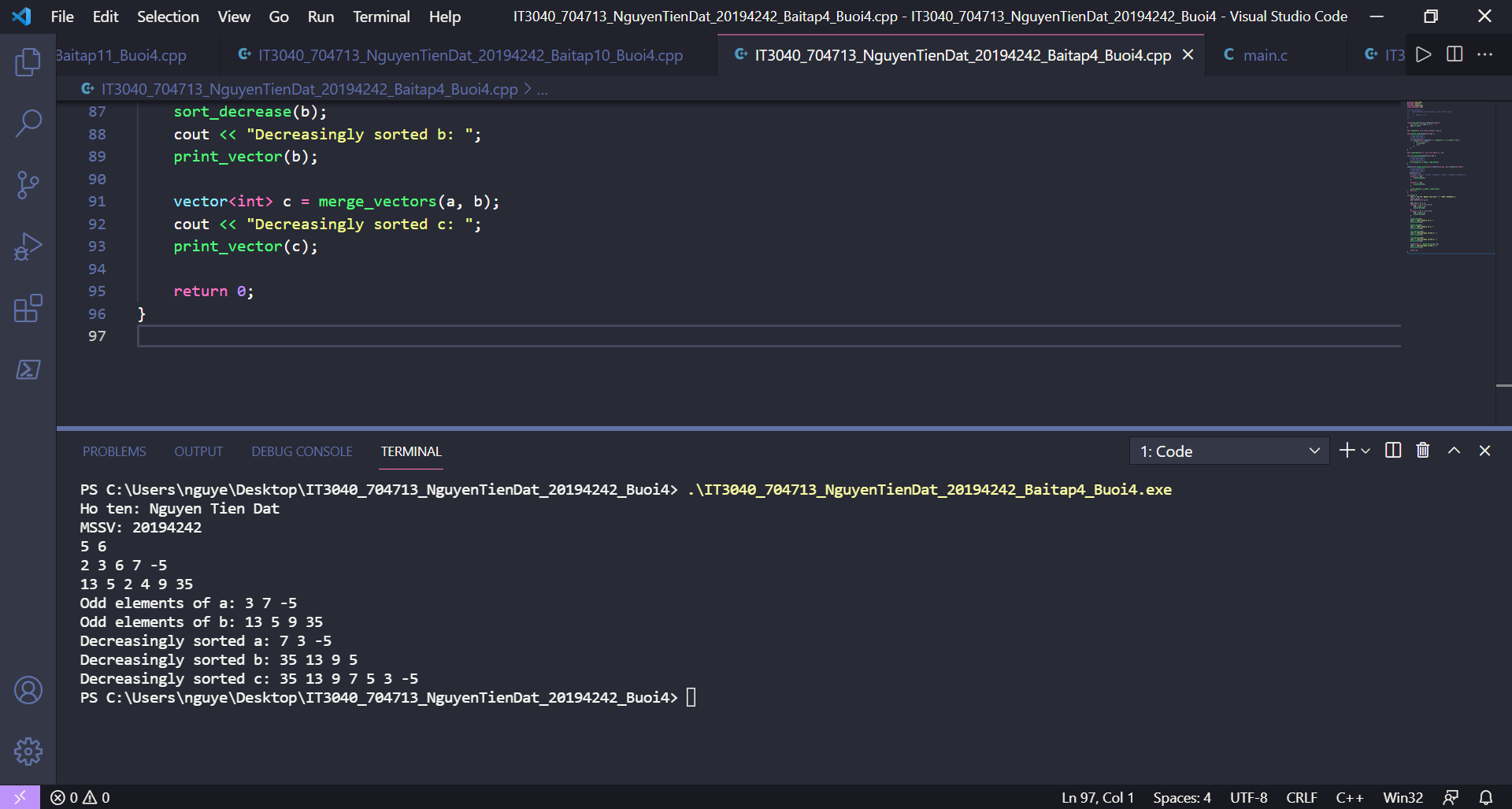
    cout << "Decreasingly sorted c: ";

    print\_vector(c);

    return 0;

}

Kết quả:



### Bài tập 5:

### Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > . Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap5\_Buoi4.cpp

#include <iostream>

#include <stack>

#include <vector>

#include <list>

using namespace std;

void dfs(vector< list<int> > *adj*) {

    stack<int> S;

    vector<bool> visited(*adj*.size());

    S.push(1); // Bắt đầu từ đỉnh số 1

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   while (!S.empty()) {

       int u = S.top();

       if (!visited[u]) {

           visited[u] = true;

           cout << u << endl;

       }

       if (!*adj*[u].empty()) {

           int v = *adj*[u].front(); // duyet tu trai sang phai

*adj*[u].pop\_front();

           if (!visited[v]) {

               S.push(v);

           }

       }

       else S.pop(); // khi ma adj[u] rong!

   }

}

int main(){

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    int n = 7;

    vector< list<int> > adj;

    adj.resize(n + 1);

    adj[1].push\_back(2);

    adj[2].push\_back(4);

    adj[1].push\_back(3);

    adj[3].push\_back(4);

    adj[3].push\_back(5);

    adj[5].push\_back(2);

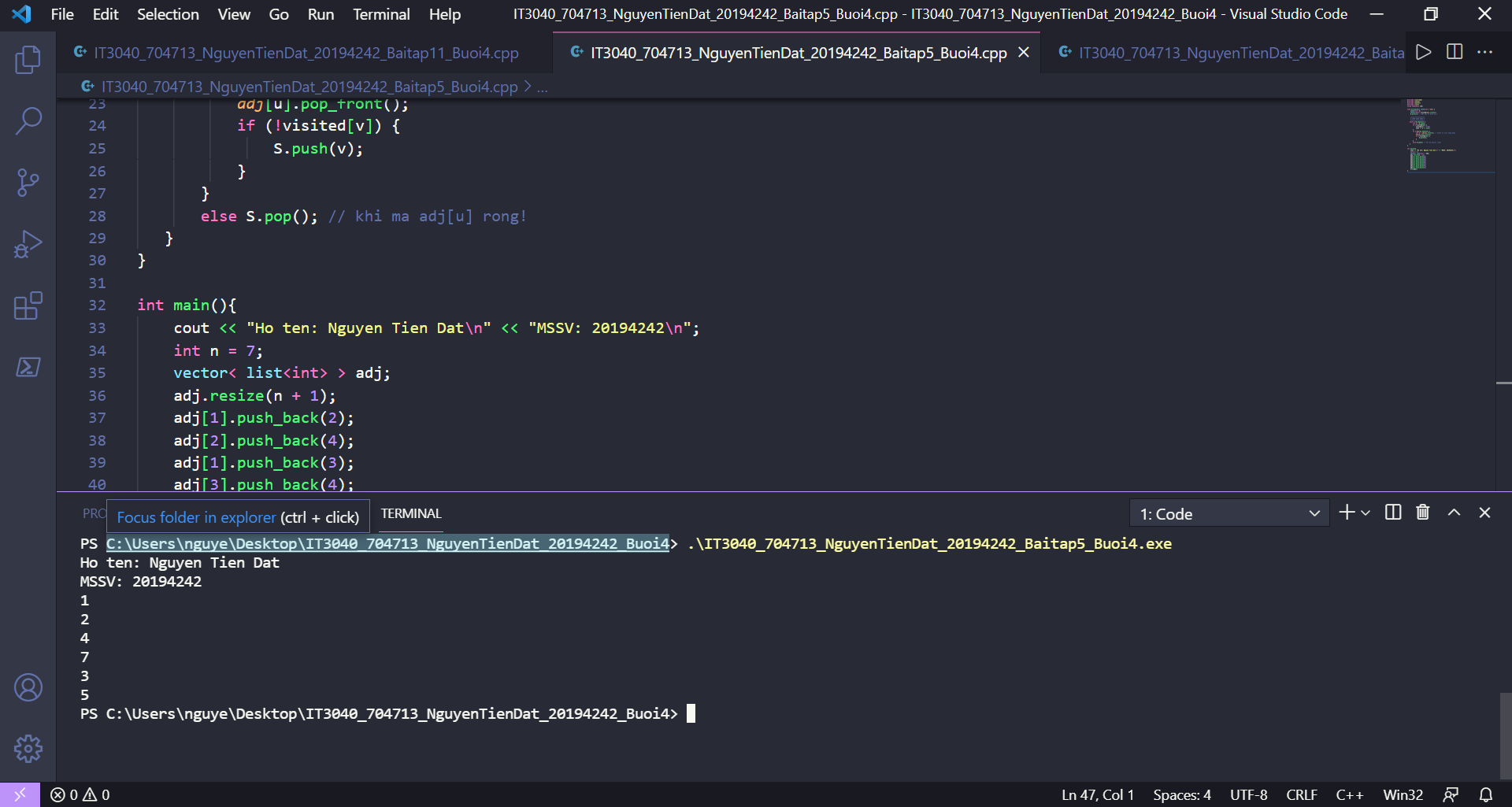
    adj[2].push\_back(7);

    adj[6].push\_back(7);

    dfs(adj);

}

Kết quả:



### Bài tập 6:

### Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > . Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap6\_Buoi4.cpp

#include <iostream>

#include <vector>

#include <list>

#include <queue>

using namespace std;

void bfs(vector< list<int> > *adj*) {

    queue<int> Q;

    vector<bool> visited(*adj*.size());

    Q.push(1); // Bắt đầu từ đỉnh số 1

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   while (!Q.empty())

   {

       int u = Q.front();

       if (!visited[u]) {

           visited[u] = true;

           cout << u << endl;

       }

       if (!*adj*[u].empty()) {

           int v = *adj*[u].front(); // duyet tu trai sang phai

*adj*[u].pop\_front();

           if (!visited[v]) {

               Q.push(v);

           }

       }

       else Q.pop(); // khi ma adj[u] rong!

   }

}

int main(){

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    int n = 7;

    vector< list<int> > adj;

    adj.resize(n + 1);

    adj[1].push\_back(2);

    adj[2].push\_back(4);

    adj[1].push\_back(3);

    adj[3].push\_back(4);

    adj[3].push\_back(5);

    adj[5].push\_back(2);

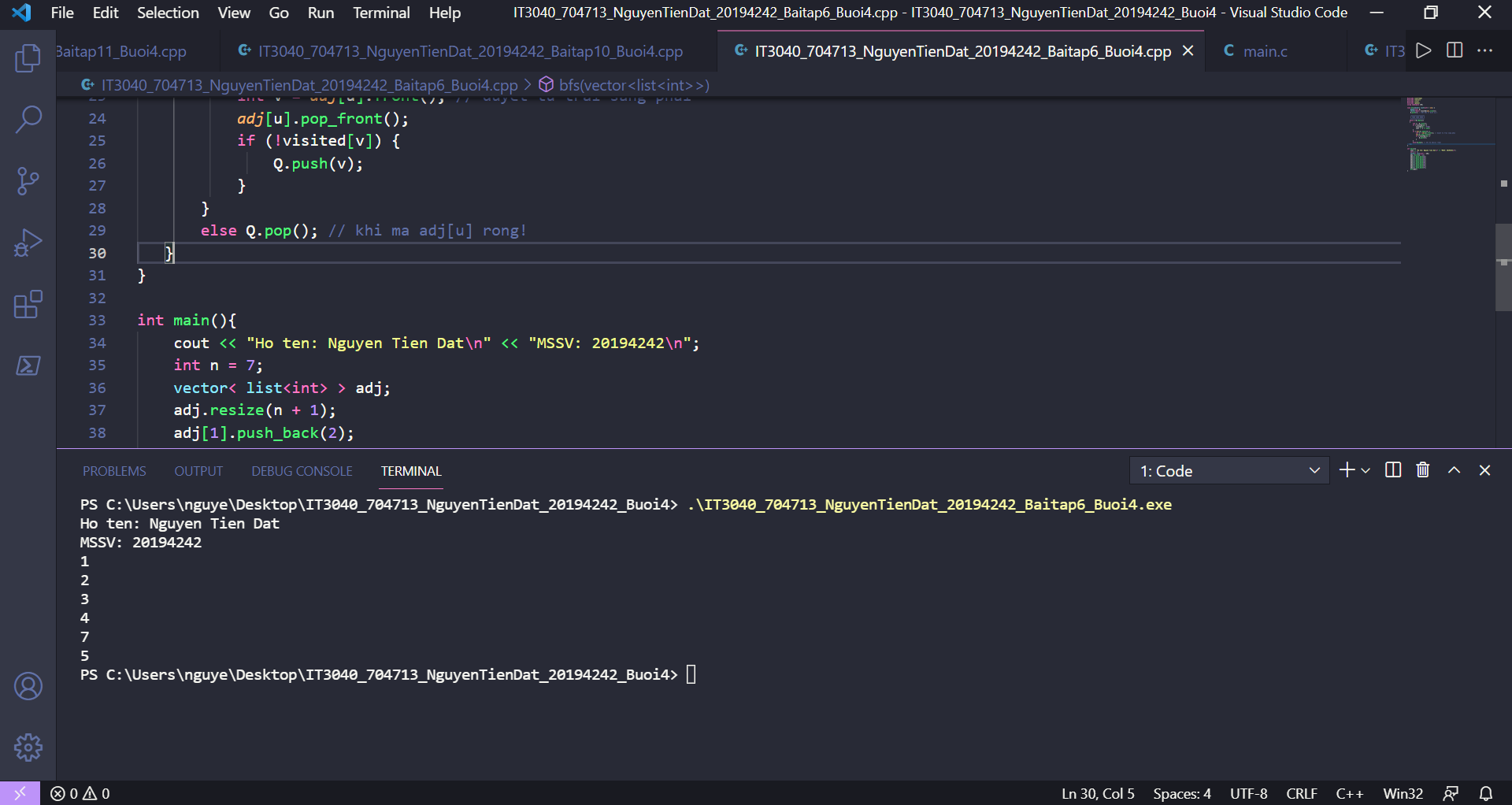
    adj[2].push\_back(7);

    adj[6].push\_back(7);

    bfs(adj);

}

Kết quả:



### Bài tập 7:

### Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap7\_Buoi4.cpp

#include <iostream>

#include <set>

using namespace std;

template<class *T*>

set<*T*> set\_union(const set<*T*> &*a*, const set<*T*> &*b*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   set<*T*> res;

   for (*T* i : *a*) {

       res.insert(i);

   }

   for (*T* i : *b*) {

       res.insert(i);

   }

   return res;

}

template<class *T*>

set<*T*> set\_intersection(const set<*T*> &*a*, const set<*T*> &*b*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   set<*T*> res;

   for (*T* i : *a*) {

       bool flag = false;

       for (*T* j : *b*) {

           if (i == j) flag = true;

       }

       if (flag) {

           res.insert(i);

       }

   }

   return res;

}

template<class *T*>

void print\_set(const std::set<*T*> &*a*) {

    for (const *T* &x : *a*) {

        std::cout << x << ' ';

    }

    std::cout << std::endl;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    std::set<int> a = {1, 2, 3, 5, 7};

    std::set<int> b = {2, 4, 5, 6, 9};

    std::set<int> c = set\_union(a, b);

    std::set<int> d = set\_intersection(a, b);

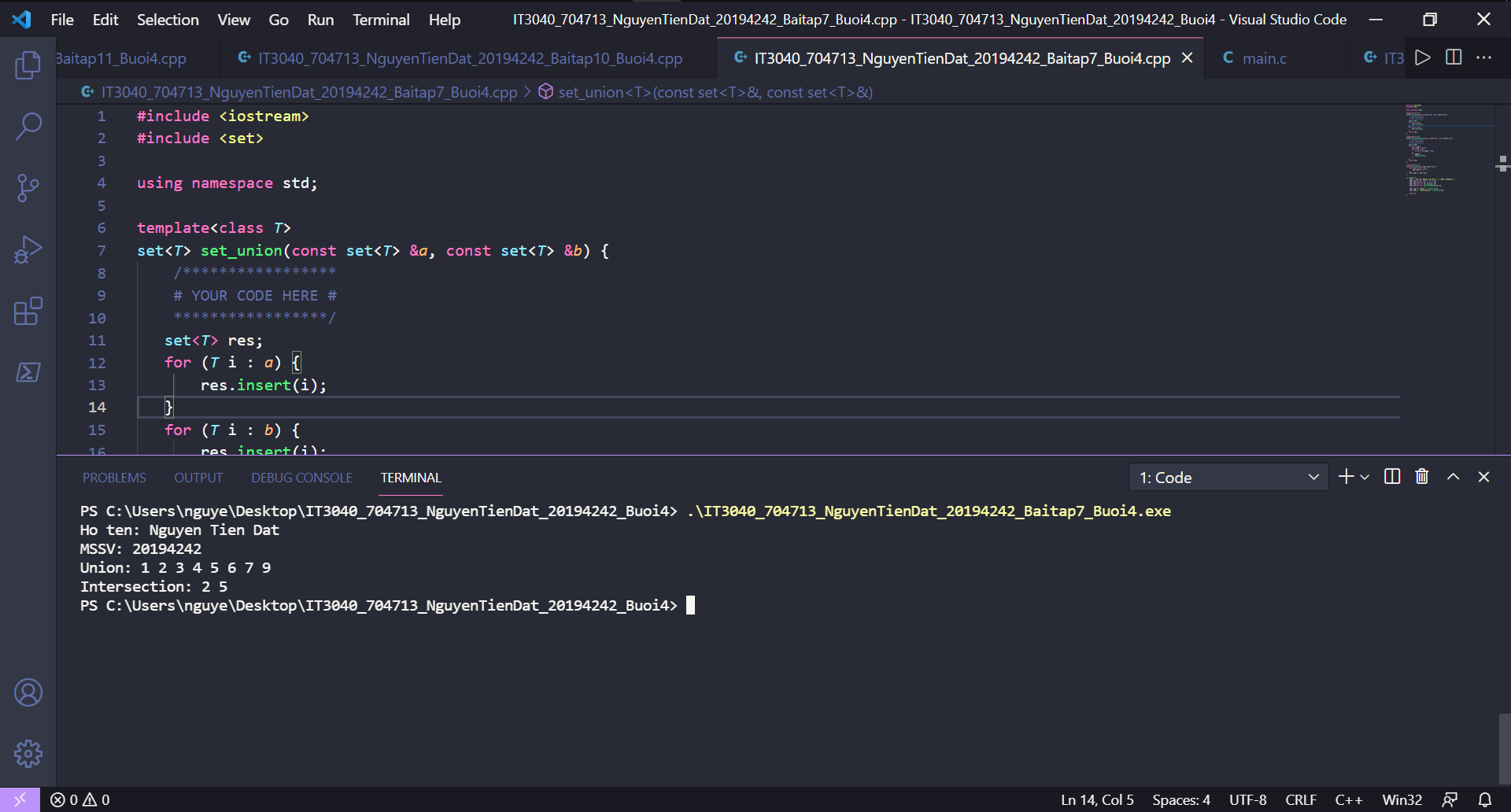
    std::cout << "Union: "; print\_set(c);

    std::cout << "Intersection: "; print\_set(d);

    return 0;

}

Kết quả:



### Bài tập 8:

### Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.

### Trong đó mỗi phần tử được gán cho một số thực trong đoạn [0..1] biểu thị độ thuộc của phần tử trong tập hợp, với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 0 nghĩa là phần tử chắc chắn không thuộc trong tập hợp.

### Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp, với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap8\_Buoi4.cpp

#include <iostream>

#include <map>

using namespace std;

template<class *T*>

map<*T*, double> fuzzy\_set\_union(const map<*T*, double> &*a*, const map<*T*, double> &*b*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   map<*T*, double> res;

   for (auto i : *a*) {

       bool flag = false;

       for (auto j : *b*) {

           if (i.first == j.first) {

               flag = true;

               res.insert({i.first, max(i.second, j.second)});

                break;

           }

       }

       if (!flag) res.insert({i.first, i.second});

   }

   for (auto i : *b*) {

       bool flag = false;

       for (auto j : *a*) {

           if (i.first == j.first) {

               flag = true;

               res.insert({i.first, max(i.second, j.second)});

                break;

           }

       }

       if (!flag) res.insert({i.first, i.second});

   }

   return res;

}

template<class *T*>

map<*T*, double> fuzzy\_set\_intersection(const map<*T*, double> &*a*, const map<*T*, double> &*b*) {

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

   map<*T*, double> res;

   for (auto i : *a*) {

       for (auto j : *b*) {

           if (i.first == j.first) {

               res.insert({i.first, min(i.second, j.second)});

           }

       }

   }

   return res;

}

template<class *T*>

void print\_fuzzy\_set(const std::map<*T*, double> &*a*) {

    cout << "{ ";

    for (const auto &x : *a*) {

        std::cout << "(" << x.first << ", " << x.second << ") ";

    }

    cout << "}";

    std::cout << std::endl;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    std::map<int, double> a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}};

    std::map<int, double> b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}};

    std::cout << "A = "; print\_fuzzy\_set(a);

    std::cout << "B = "; print\_fuzzy\_set(b);

    std::map<int, double> c = fuzzy\_set\_union(a, b);

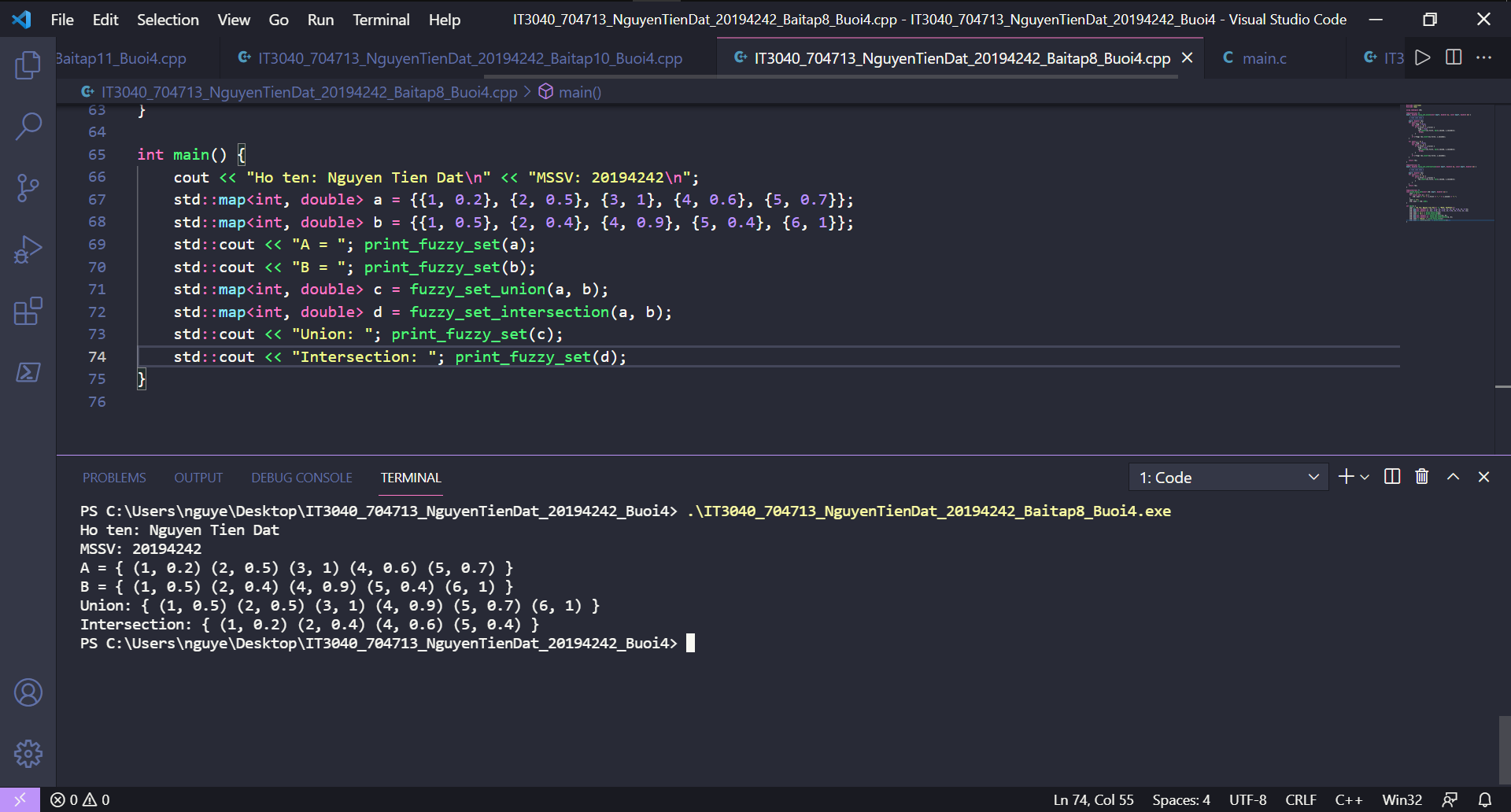
    std::map<int, double> d = fuzzy\_set\_intersection(a, b);

    std::cout << "Union: "; print\_fuzzy\_set(c);

    std::cout << "Intersection: "; print\_fuzzy\_set(d);

}

Kết quả:



### Bài tập 9:

### Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng priority\_queue Cụ thể, bạn cần cài đặt hàm vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) nhận đầu vào là danh sách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng số tương ứng của cạnh. Đồ thị gồm n đỉnh được đánh số từ 0 tới n-1. Hàm cần trả vector<int> chứa n phần tử lần lượt là khoảng cách đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ..., n-1.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap9\_Buoi4.cpp

#include <iostream>

#include <queue>

#include <climits>

using namespace std;

struct *compare* {

    // sap xep queue tang dan

    // priority\_queue lay ra phan tu co do uu tien cao nhat (phan tu nho nhat)

    bool operator() (pair<int, int> *l*, pair<int, int> *r*) {

        return *l*.second > *r*.second;

    }

};

vector<int> dijkstra(const vector< vector< pair<int, int> > >&*adj*) {

   priority\_queue<pair<int, int>, vector<pair<int, int>>, *compare*> pQ;

   vector<int> dist = vector<int>(*adj*.size()); // k/c tu dinh 0 den cac dinh i (i = 0..n)

   // ban dau cac dist rat lon

    for (unsigned int i = 0; i < dist.size(); i++) {

        dist[i] = INT\_MAX;

    }

    // k/c tu 0 den 0 la 0

    dist[0] = 0;

    // khoi tao priority\_queue

    for (unsigned int i = 0; i < *adj*.size(); i++) {

        pQ.push({i, dist[i]});

    }

    while (!pQ.empty()) {

        pair<int, int> u = pQ.top(); pQ.pop();

        for (pair<int, int> v : *adj*[u.first]) {

            if (dist[v.first] > dist[u.first] + v.second) {

                dist[v.first] = dist[u.first] + v.second;

                pQ.push({v.first, dist[v.first]});

            }

        }

    }

    return dist;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    int n = 9;

    vector< vector< pair<int, int> > > adj(n);

    auto add\_edge = [&*adj*] (int *u*, int *v*, int *w*) {

        adj[*u*].push\_back({*v*, *w*});

        adj[*v*].push\_back({*u*, *w*});

    };

    add\_edge(0, 1, 4);

    add\_edge(0, 7, 8);

    add\_edge(1, 7, 11);

    add\_edge(1, 2, 8);

    add\_edge(2, 3, 7);

    add\_edge(2, 8, 2);

    add\_edge(3, 4, 9);

    add\_edge(3, 5, 14);

    add\_edge(4, 5, 10);

    add\_edge(5, 6, 2);

    add\_edge(6, 7, 1);

    add\_edge(6, 8, 6);

    add\_edge(7, 8, 7);

    vector<int> distance = dijkstra(adj);

    for (int i = 0; i < distance.size(); ++i) {

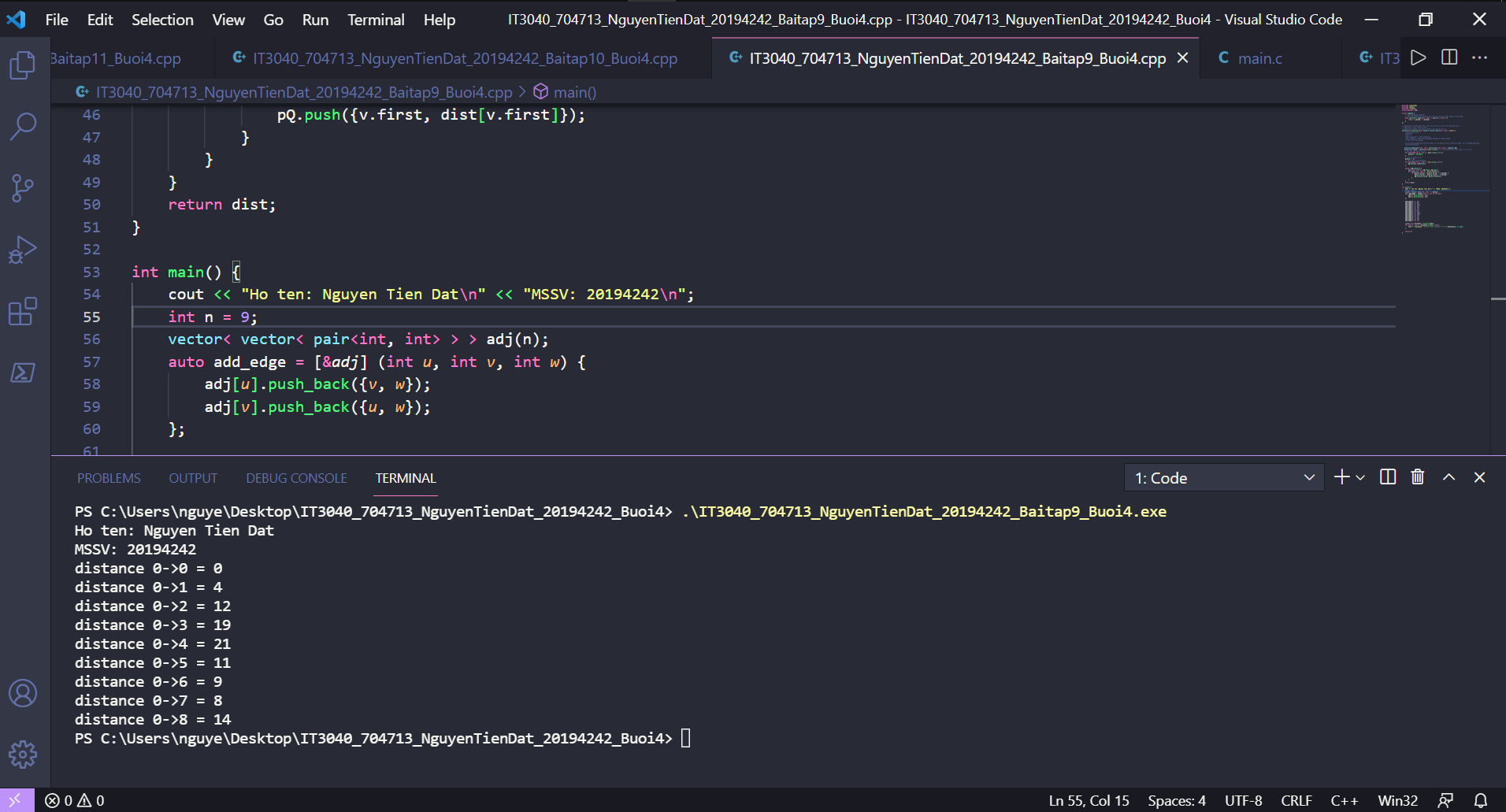
        cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;

    }

    return 0;

}

Kết quả:



## Phần 2: Bài tập về nhà

### Bài tập 10: Search Engine

### Xây dựng một máy tìm kiếm (search engine) đơn giản.

### Cho NN văn bản và QQ truy vấn. Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất.

### Sử dụng phương pháp tính điểm TF-IDF

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap10\_Buoi4.cpp

#include <iostream>

#include <map>

#include <utility>

#include <vector>

#include <sstream>

#include <algorithm>

#include <cmath>

using namespace std;

int n, q;

vector< vector<*string*> > txt;

vector< vector<*string*> > query;

map< pair<*string*, int>, int > f; // so lan xuat hien pair (tu t trong vb thu d)

vector<int> max\_f; // so lan xuat hien nhieu nhau cua tu trong vb thu d

map< *string*, int > df;// so van ban chua tu t

map< pair<*string*, int>, double > score; // diem cua tu t trong van ban thu d la kieu double

vector<*string*> split(const *string* &*p*) {

    vector<*string*> res;

*stringstream* ss(*p*);

    while (ss.good()) {

*string* substr;

        getline(ss, substr, ',');

        res.push\_back(substr);

    }

    return res;

}

void in(const vector<*string*> &*p*) {

    for (auto &it : *p*) {

        cout << it << " ";

    }

}

void input() {

*string* enter;

    cin >> n;

    getline(cin, enter); // cin.ignore()

    for (int i = 0; i < n; i++) {

*string* tmp;

        getline(cin, tmp);

        txt.push\_back(split(tmp));

    }

    cin >> q;

    getline(cin, enter); // cin.ignore()

    for (int i = 0; i < q; i++) {

*string* tmp;

        getline(cin, tmp);

        query.push\_back(split(tmp));

    }

}

// tinh so lan xuat hien cua tu t trong van ban d

int fCal(*string* *t*, int *d*) {

    if (f.find({*t*, *d*}) != f.end())  // tuc la {t,d} ton tai trong f

        return f[{*t*,*d*}]; // f.find({t,d})->second;

    // else

    int count = 0;

    for (unsigned int i = 0; i < txt[*d*].size(); i++) {

        if (!*t*.compare(txt[*d*][i])) {

            count++;

        }

    }

    f.insert({{*t*, *d*}, count});

    return count;

}

// tinh so lan lap lai nhieu nhat trong van ban d

void max\_fCal() {

    for (int i = 0; i < n; i++) {

        map<*string*, int> mini\_dict;

        int maxf = 0;

        // cal mini\_dict

        for (unsigned int j = 0; j < txt[i].size(); j++) {

            // neu da ton tai trong mini\_dict

            auto it = mini\_dict.find(txt[i][j]);

            if (it != mini\_dict.end()) {

                it->second++;

            }

            else {

                mini\_dict.insert({txt[i][j], 1});

            }

        }

        for (auto it : mini\_dict) {

            maxf = max(maxf, it.second);

        }

        max\_f.push\_back(maxf);

    }

}

// tinh so van ban chua tu t

int dfCal(*string* *t*) {

    // neu tu t da co trong df r

    if (df.find(*t*) != df.end()) {

        return df[*t*]; // df.find(t)->second;

    }

    // else

    int count = 0;

    for (auto it : txt) {

        vector<*string*>::*iterator* it\_element = find(it.begin(), it.end(), *t*);

        if (it\_element != it.end()) count++;

    }

    df.insert({*t*, count});

    return count;

}

bool check(*string* *t*, int *d*) {

    return (txt[*d*].end() != find(txt[*d*].begin(), txt[*d*].end(), *t*));

}

double scoreCal(*string* *t*, int *d*) {

    if (score.find({*t*,*d*}) != score.end()) return score[{*t*,*d*}];

    // else

    double score\_res = 0;

    if (check(*t*,*d*)) {

        int f\_cal = fCal(*t*,*d*);

        int max = max\_f[*d*];

        int df\_cal = dfCal(*t*);

        double TF = 0.5 + 0.5 \* ((double)f\_cal / max); //cout << "TF: " << TF << endl;

        double IDF = log2((double)n / df\_cal); //cout << "IDF: " << IDF << endl;

        score\_res = TF \* IDF;

    }

    score.insert({{*t*,*d*}, score\_res});

    return score\_res;

}

int searchResult(vector<*string*> *query*) {

    int res = 0;

    double max\_score = 0;

    for (int i = 0; i < n; i++) {

        double score = 0;

        for (unsigned int j = 0; j < *query*.size(); j++) {

            score += scoreCal(*query*[j], i);

        }

        if (max\_score < score) {

            max\_score = score;

            res = i;

        }

    }

    return res + 1;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

*ios\_base*::sync\_with\_stdio(false); cin.tie(NULL);

    input();

    // tinh maxf

    max\_fCal();

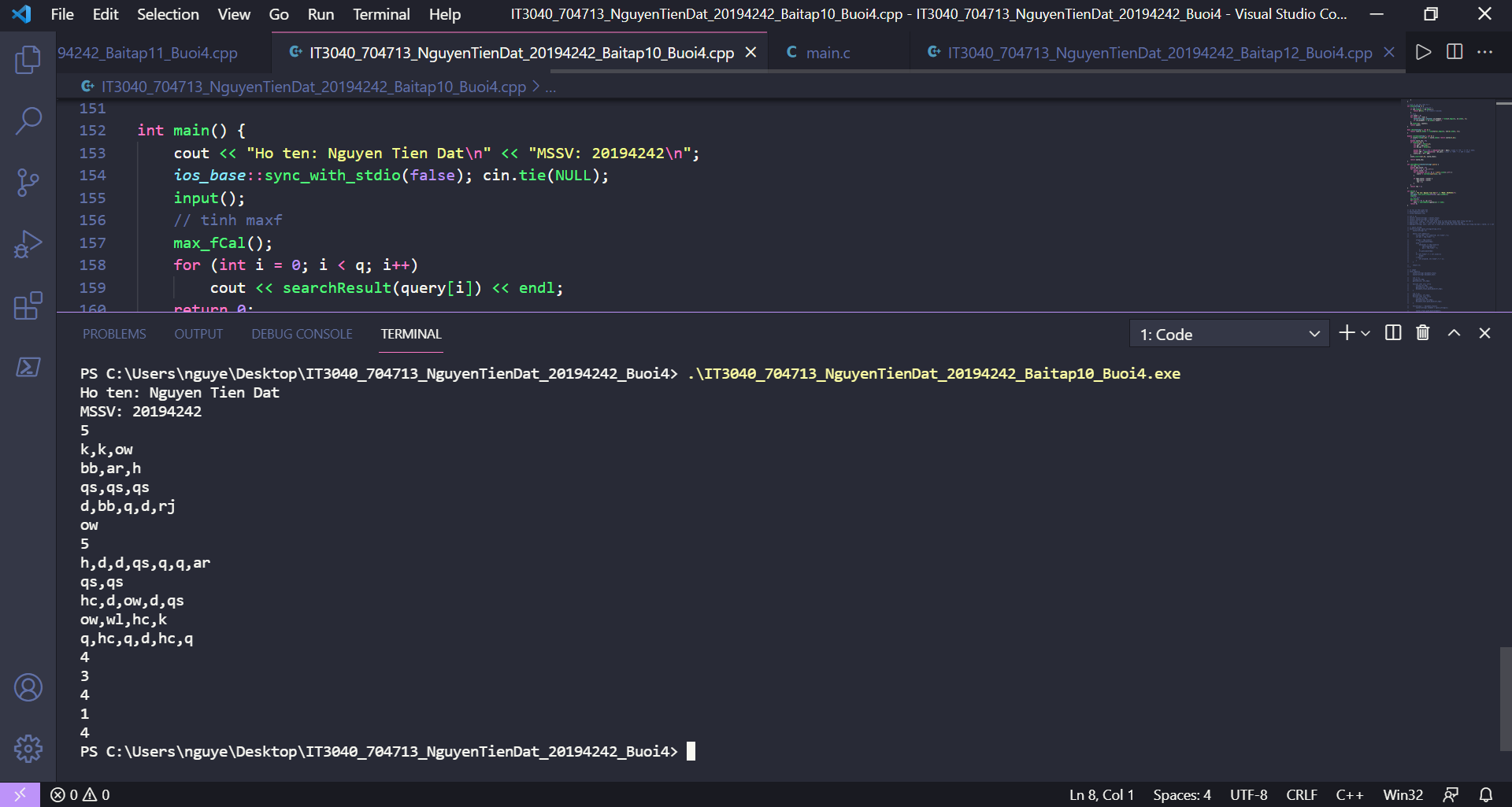
    for (int i = 0; i < q; i++)

        cout << searchResult(query[i]) << endl;

    return 0;

}

Kết quả:



### Bài tập 11. Bảo vệ lâu đài

### Bức tường bao quanh một lâu đài nọ được cấu thành từ nn đoạn tường được đánh số từ 1 đến nn. Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi aiai tên giặc đánh vào đoạn tường thứ ii. Để bảo vệ lâu đài có tất cả ss lính.

### Do các đoạn tường có chất lượng khác nhau nên khả năng bảo vệ tại các đoạn tường cũng khác nhau. Cụ thể tại đoạn tường thứ ii, mỗi lính có thể đẩy lùi tấn công của kiki tên giặc.

### Giả sử đoạn tường thứ ii có xixi lính. Khi đó nếu số tên giặc không vượt quá xi×kixi×ki thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai−xi×kiai−xi×ki tên giặc lọt vào lâu đài qua đoạn tường này.

### Yêu cầu hãy viết chương trình phân bố lính đứng ở các đoạn tường sao cho tổng số lính là ss và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap11\_Buoi4.cpp

#include <iostream>

#include <vector>

#include <queue>

#include <utility>

using namespace std;

// y tuong la dung priority queue, moi lan pop ra la 1 linh

// ai la so ke dich cua 1 doan tuong i : pair.first

// ki la kha nang chong dich cua 1 doan tuong i: pair.second

struct *compare* {

    bool operator() (pair<int, int> *lhs*, pair<int, int> *rhs*) {

        int l, r;

        // do ai va ki triet tieu cho nhau nen chon phan tu nho hon!!!

        if (*lhs*.first < *lhs*.second) l = *lhs*.first;

        else l = *lhs*.second;

        if (*rhs*.first < *rhs*.second) r = *rhs*.first;

        else r = *rhs*.second;

        return l < r; // less -> sx priority queue giam dan, cang cao cang uu tien

    }

};

int n, s;

vector <pair<int, int>> in;

priority\_queue < pair<int, int>, vector <pair<int, int>>, *compare* > pq;

int total\_enemy = 0, killed\_enemy = 0;

void input() {

    cin >> n >> s;

    int ai, ki;

    for (int i = 0; i < n; i++) {

        cin >> ai >> ki;

        in.push\_back({ai, ki});

        pq.push({ai, ki});

        total\_enemy += ai;

    }

}

void solve() {

    while (!pq.empty() && s > 0) {

        auto it = pq.top(); pq.pop();

        // neu 1 linh chap het ki giac

        if (it.first < it.second) killed\_enemy += it.first;

        else {

            killed\_enemy += it.second \* 1; // 1 linh ok?

            int enemy\_conlai = it.first - it.second \* 1; // tinh so giac con lai

            pq.push({enemy\_conlai, it.second}); // so giac con lai cho vao cuoi hang doi, neu con linh thi xu ly tiep

        }

        s--; // dung 1 linh trong 1 lan lap

    }

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

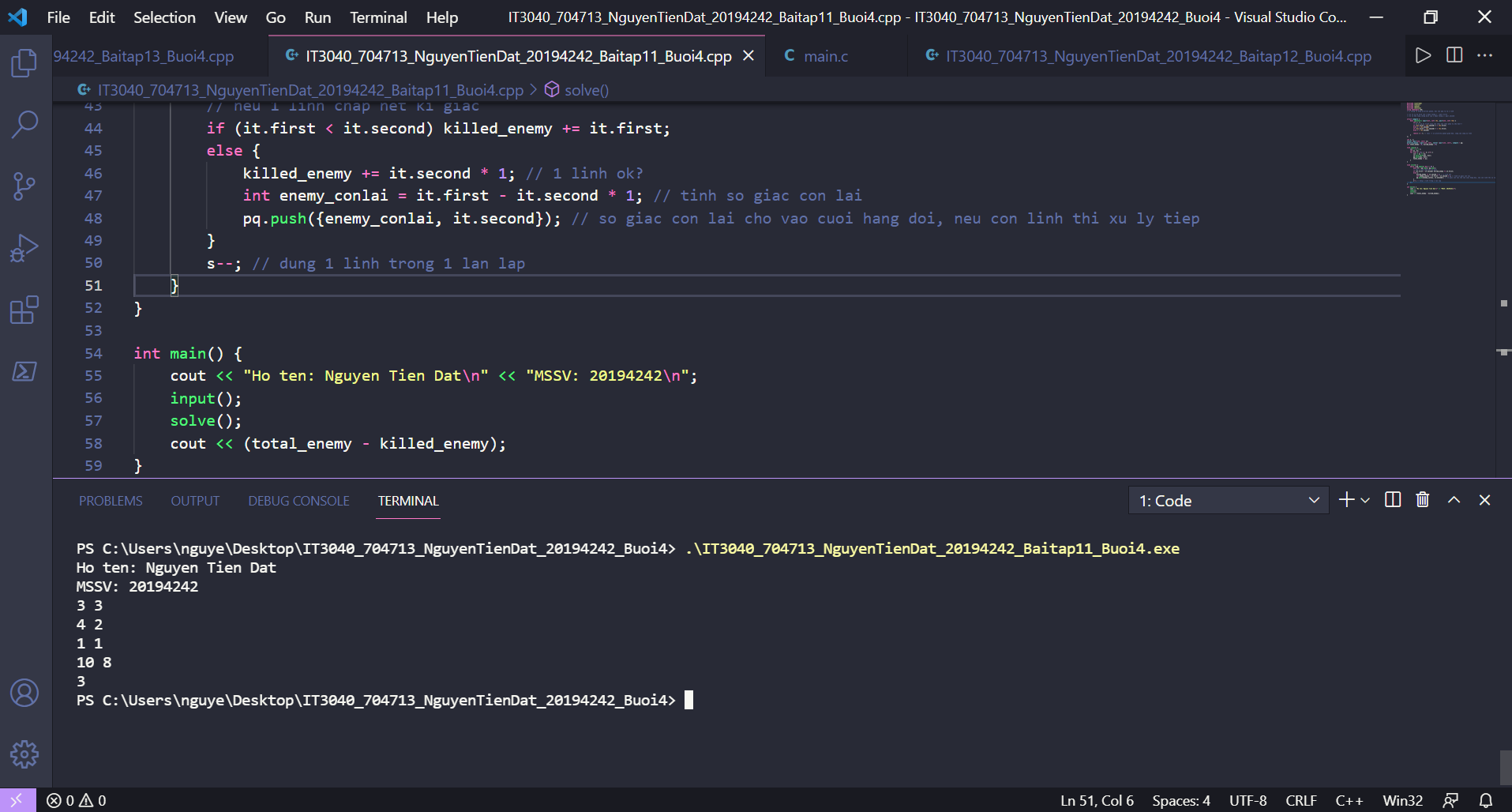
    input();

    solve();

    cout << (total\_enemy - killed\_enemy);

}

Kết quả:



### Bài tập 12. Lược đồ

### Cho một lược đồ gồm nn cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hnh1,h2,…,hn. Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap12\_Buoi4.cpp

 #include <iostream>

 #include <vector>

 using namespace std;

 vector<int> in;

 int n;

 int area = 0;

 int max\_area = 0;

 void input() {

    cin >> n;

    int tmp;

    for (int i = 0; i < n; i++) {

        cin >> tmp;

        in.push\_back(tmp);

    }

 }

 void solve() {

    for (int i = 0; i < n; i++) {

        int j = i, k = i;

        while (in[j-1] >= in[i] && j > 0) j--;

        while (in[k+1] >= in[i] && k < n-1) k++;

        area = in[i] \* (k-j+1);

        if (area > max\_area) max\_area = area;

    }

 }

 int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

    input();

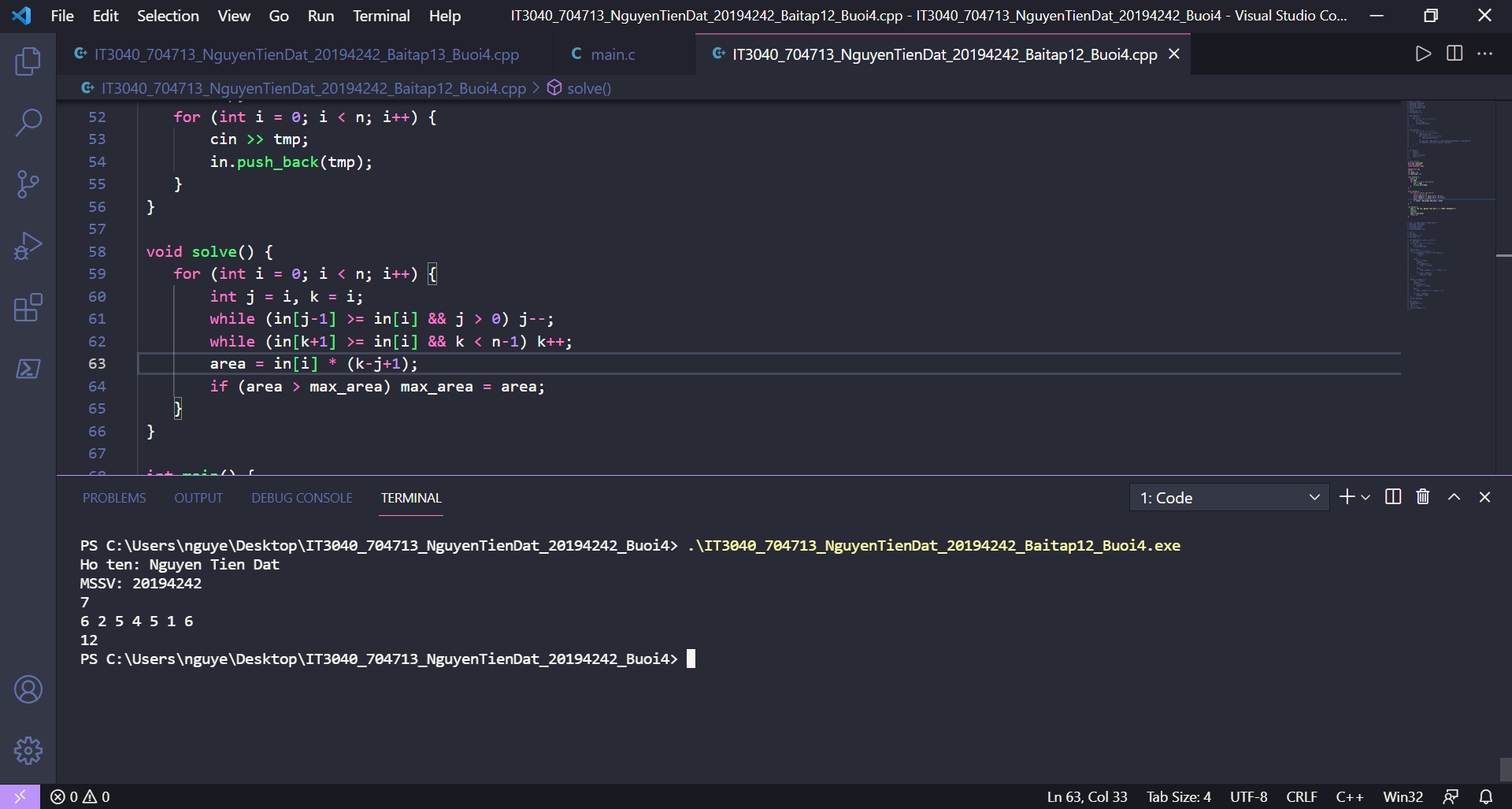
    solve();

    cout << max\_area;

    return 0;

 }

Kết quả:



### Bài tập 13: Đếm xâu con

### Cho một xâu nhị phân độ dài nn. Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap13\_Buoi4.cpp

#include <iostream>

#include <stdio.h>

#include <map>

using namespace std;

// sum\_difference la do chech lech giua 0 va 1 tai thoi diem i

// F[sum\_difference] la so lan xuat hien do chech lech sum\_difference do

map<int, unsigned int> F;

int sum\_differenece = 0;

int solve(*string* *s*) {

    unsigned int count = 0;

    int n = *s*.size();

    for (int i = -n-1; i < n+1; i++) {

        F.insert({i,0});

    }

    F[0] = 1;

    for (int i = 0; i < n; i++) {

        if (*s*[i] == '0') {

            sum\_differenece--;

        }

        else sum\_differenece++;

        count += F[sum\_differenece];

        ++F[sum\_differenece];

    }

    return count;

}

int main() {

    cout << "Ho ten: Nguyen Tien Dat\n" << "MSSV: 20194242\n";

*string* s; cin >> s;

    cout << solve(s);

}

Kết quả:

