Contents

[Bài thực hành số 3 2](#_Toc153123687)

[**Bài 4.1:**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)) 2](#_Toc153123688)

[**Bài 4.2:** Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm tính diện tích tam giác theo tọa độ 3 đỉnh. 8](#_Toc153123689)

[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. 9](#_Toc153123690)

[Bài 4.4. Cho hai std::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. 11](#_Toc153123691)

[Bài 4.5. Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. 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). 17](#_Toc153123692)

[Bài 4.6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. 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). 20](#_Toc153123693)

[Bài 4.7. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set 23](#_Toc153123694)

[Bài 4.8. 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. 27](#_Toc153123695)

[Bài 4.9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::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. 30](#_Toc153123696)

[Bài tập 4.10: Search Engine Xây dựng một máy tìm kiếm (search engine) đơn giản. Cho N văn bản và Q 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: 35](#_Toc153123697)

[Bài tập 4.11. Bảo vệ lâu đài Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n . Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi ai tên giặc đánh vào đoạn tường thứ i . Để bảo vệ lâu đài có tất cả s 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 i , mỗi lính có thể đẩy lùi tấn công của ki tên giặc. Giả sử đoạn tường thứ i có xi lính. Khi đó nếu số tên giặc không vượt quá xi×ki thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai−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 s và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất. 46](#_Toc153123698)

[Bài tập 4.12. Lược đồ Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp. 50](#_Toc153123699)

[Bài tập 4.13: Đếm xâu con Cho một xâu nhị phân độ dài n . Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau. 54](#_Toc153123700)

[Hình 1. Code bài 4.1 ảnh 1 6](#_Toc153123701)

[Hình 2. Code bài 4.1 ảnh 2 6](#_Toc153123702)

[Hình 3. Code bài 4.1 ảnh 3 7](#_Toc153123703)

[Hình 4. Code bài 4.1 ảnh 4 7](#_Toc153123704)

[Hình 5. Code bài 4.1 ảnh 5 8](#_Toc153123705)

[Hình 6. Code bài 4.1 ảnh 6 8](#_Toc153123706)

[Hình 7. Test bài 4.1 8](#_Toc153123707)

[Hình 8. Code bài 4.2 ảnh 1 12](#_Toc153123708)

[Hình 9. Code bài 4.2 ảnh 2 12](#_Toc153123709)

[Hình 10. Test bài 4.2 13](#_Toc153123710)

[Hình 11. Code bài 4.3 ảnh 1 14](#_Toc153123711)

[Hình 12. Code bài 4.3 ảnh 2 14](#_Toc153123712)

[Hình 13. Test bài 4.3 14](#_Toc153123713)

[Hình 14. Code bài 4.4 ảnh 1 15](#_Toc153123714)

[Hình 15. Code bài 4.4 ảnh 2 16](#_Toc153123715)

[Hình 16. Code bài 4.4 ảnh 3 16](#_Toc153123716)

[Hình 17. Code bài 4.4 ảnh 4 17](#_Toc153123717)

[Hình 18. Code bài 4.4 ảnh 5 17](#_Toc153123718)

[Hình 19. Test bài 4.4 17](#_Toc153123719)

[Hình 20. Code bài 4.5 ảnh 1 21](#_Toc153123720)

[Hình 21. Code bài 4.5 ảnh 2 21](#_Toc153123721)

[Hình 22. Code bài 4.5 ảnh 3 22](#_Toc153123722)

[Hình 23. Test bài 4.5 ảnh 1 22](#_Toc153123723)

[Hình 24. Test bài 4.5 ảnh 2 23](#_Toc153123724)

[Hình 25. Code bào 4.6 ảnh 1 24](#_Toc153123725)

[Hình 26. Code bào 4.6 ảnh 2 25](#_Toc153123726)

[Hình 27. Test bài 4.6 ảnh 1 25](#_Toc153123727)

[Hình 28. Test bài 4.6 ảnh 2 26](#_Toc153123728)

[Hình 29. Code bài 4.7 ảnh 1 27](#_Toc153123729)

[Hình 30. Code bài 4.7 ảnh 2 28](#_Toc153123730)

[Hình 31. Code bài 4.7 ảnh 3 28](#_Toc153123731)

[Hình 32. Code bài 4.7 ảnh 4 28](#_Toc153123732)

[Hình 33. Code bài 4.7 ảnh 5 29](#_Toc153123733)

[Hình 34. Test bài 4.7 29](#_Toc153123734)

[Hình 35. Code bài 4.8 ảnh 1 31](#_Toc153123735)

[Hình 36. Code bài 4.8 ảnh 2 31](#_Toc153123736)

[Hình 37. Code bài 4.8 ảnh 3 32](#_Toc153123737)

[Hình 38. Code bài 4.8 ảnh 4 32](#_Toc153123738)

[Hình 39. Code bài 4.8 ảnh 5 32](#_Toc153123739)

[Hình 40. Test bài 4.8 ảnh 1 33](#_Toc153123740)

[Hình 41. Test bài 4.8 ảnh 2 33](#_Toc153123741)

[Hình 42. Code bài 4.9 ảnh 1 34](#_Toc153123742)

[Hình 43. Code bài 4.9 ảnh 2 34](#_Toc153123743)

[Hình 44. Code bài 4.9 ảnh 3 35](#_Toc153123744)

[Hình 45. Code bài 4.9 ảnh 4 35](#_Toc153123745)

[Hình 46. Test bài 4.9 ảnh 1 36](#_Toc153123746)

[Hình 47. Test bài 4.9 ảnh 2 36](#_Toc153123747)

[Hình 48. Code bài 4.10 ảnh 1 39](#_Toc153123748)

[Hình 49. Code bài 4.10 ảnh 2 39](#_Toc153123749)

[Hình 50. Code bài 4.10 ảnh 3 40](#_Toc153123750)

[Hình 51. Code bài 4.10 ảnh 4 40](#_Toc153123751)

[Hình 52. Code bài 4.10 ảnh 5 40](#_Toc153123752)

[Hình 53. Code bài 4.10 ảnh 6 41](#_Toc153123753)

[Hình 54. Test case 1 bài 4.10 ảnh 1 41](#_Toc153123754)

[Hình 55. Test case 1 bài 4.10 ảnh 2 42](#_Toc153123755)

[Hình 56. Test case 2 bài 4.10 ảnh 1 42](#_Toc153123756)

[Hình 57. Test case 2 bài 4.10 ảnh 2 43](#_Toc153123757)

[Hình 58. Test case 3 bài 4.10 ảnh 1 43](#_Toc153123758)

[Hình 59. Test case 3 bài 4.10 ảnh 2 44](#_Toc153123759)

[Hình 60. Test case 4 bài 4.10 ảnh 1 44](#_Toc153123760)

[Hình 61. Test case 4 bài 4.10 ảnh 2 45](#_Toc153123761)

[Hình 62. Test case 5 bài 4.10 ảnh 1 45](#_Toc153123762)

[Hình 63. Test case 5 bài 4.10 ảnh 2 46](#_Toc153123763)

[Hình 64. Code bài 4.11 ảnh 1 50](#_Toc153123764)

[Hình 65. Code bài 4.11 ảnh 2 51](#_Toc153123765)

[Hình 66. Test case 1 bài 4.11 51](#_Toc153123766)

[Hình 67. Test case 2 bài 4.11 51](#_Toc153123767)

[Hình 68. Test case 3 bài 4.11 52](#_Toc153123768)

[Hình 69. Test case 4 bài 4.11 52](#_Toc153123769)

[Hình 70. Test case 5 bài 4.11 52](#_Toc153123770)

[Hình 71. Code bài 4.12 ảnh 1 54](#_Toc153123771)

[Hình 72. Code bài 4.12 ảnh 2 55](#_Toc153123772)

[Hình 73. Code bài 4.12 ảnh 3 55](#_Toc153123773)

[Hình 74. Test case 1 bài 4.12 55](#_Toc153123774)

[Hình 75. Test case 2 bài 4.12 56](#_Toc153123775)

[Hình 76. Test case 3 bài 4.12 56](#_Toc153123776)

[Hình 77. Test case 4 bài 4.12 56](#_Toc153123777)

[Hình 78. Code bài 4.13 ảnh 1 58](#_Toc153123778)

[Hình 79. Code bài 4.13 ảnh 2 59](#_Toc153123779)

[Hình 80. Test case 1 bài 4.13 59](#_Toc153123780)

[Hình 81. Test case 2 bài 4.13 59](#_Toc153123781)

[Hình 82. Test case 3 bài 4.13 59](#_Toc153123782)

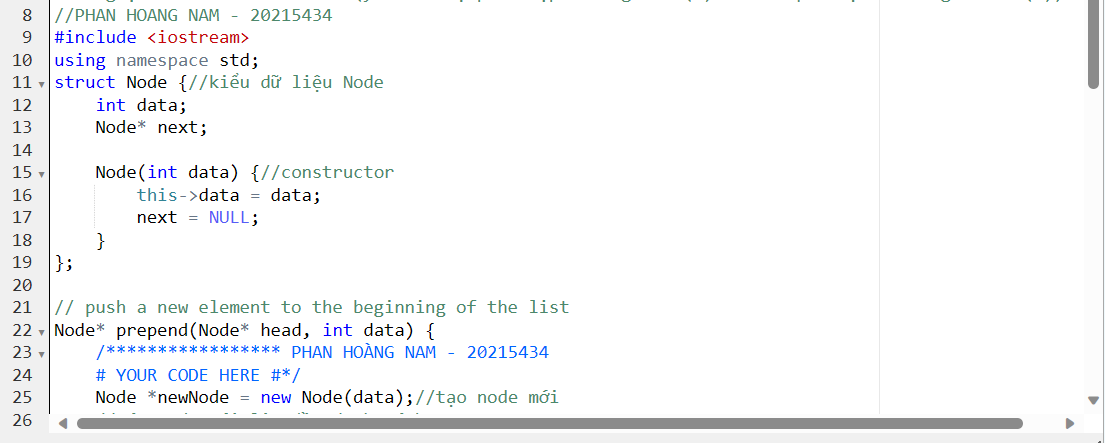
[Hình 83. Test case 4 bài 4.13 60](#_Toc153123783)

# Bài thực hành số 3

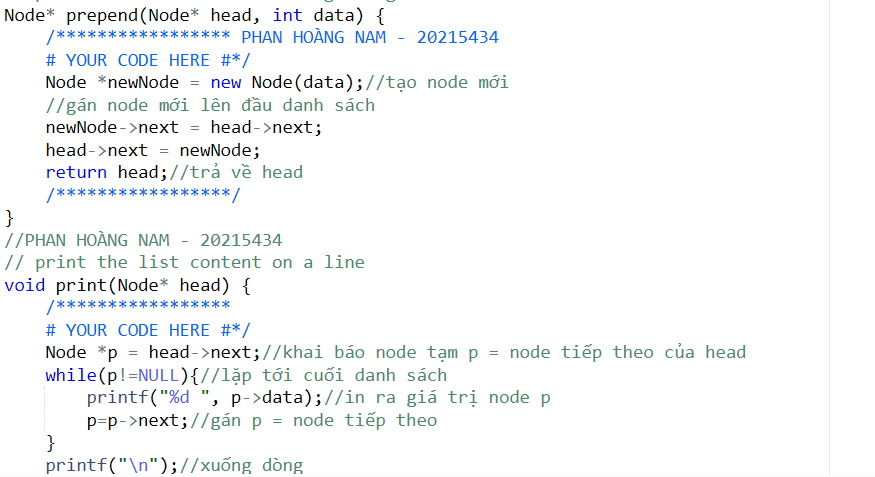
## **Bài 4.1:**Đảo ngược một danh sách liên kết đơn Hãy hoàn thiện các hàm thao tác trên một danh sách liên kết: Thêm một phần tử vào đầu danh sách liên kết In danh sách Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian O(N) và chi phí bộ nhớ dùng thêm O(1))



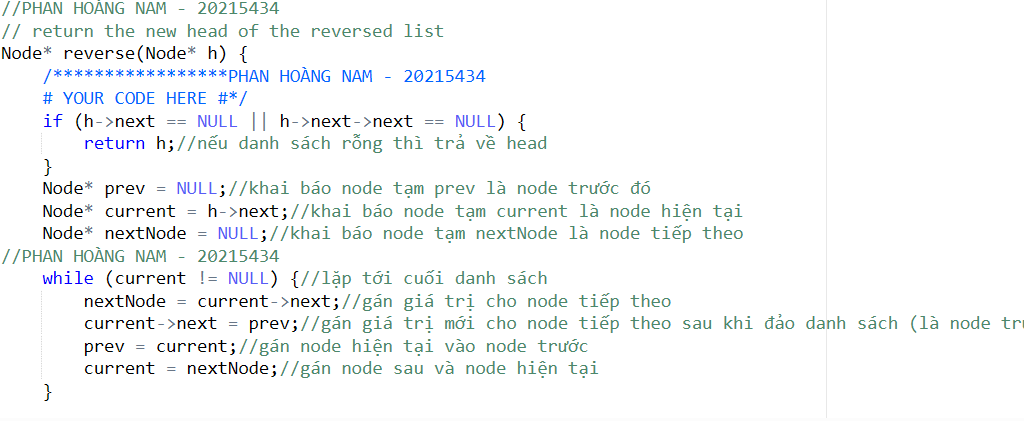
Hình 1. Code bài 4.1 ảnh 1



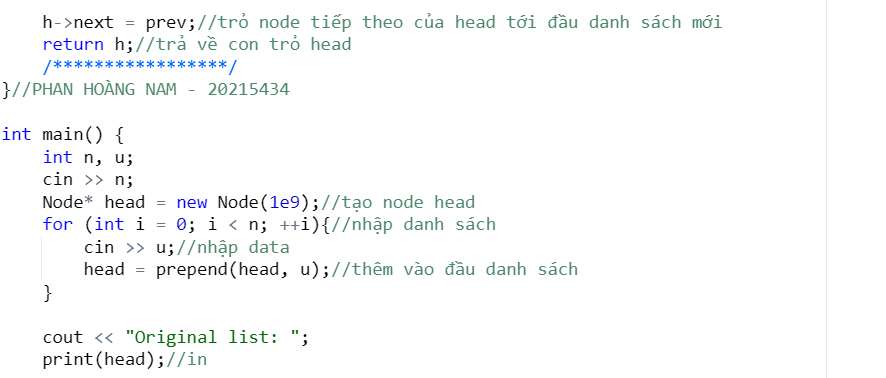
Hình 2. Code bài 4.1 ảnh 2



Hình 3. Code bài 4.1 ảnh 3



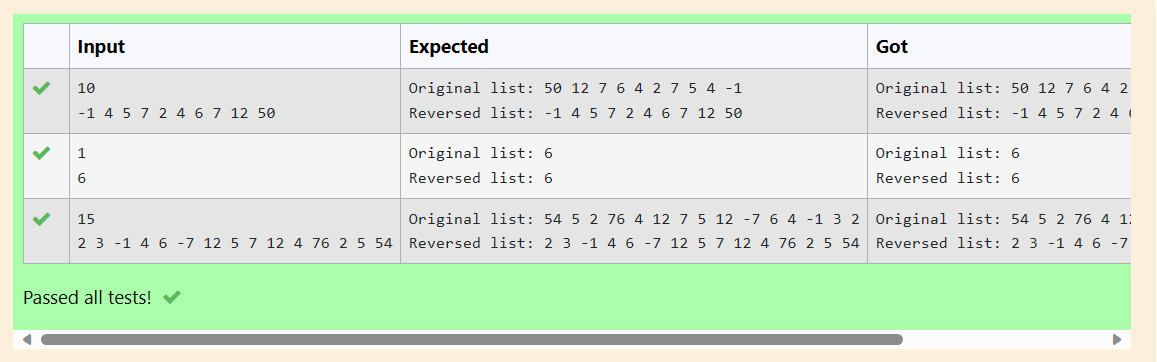
Hình 4. Code bài 4.1 ảnh 4



Hình 5. Code bài 4.1 ảnh 5



Hình 6. Code bài 4.1 ảnh 6



Hình 7. Test bài 4.1

/\*Bài 4.1: 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))\*/

//PHAN HOANG NAM - 20215434

#include <iostream>

using namespace std;

struct Node {//kiểu dữ liệu Node

int data;

Node\* next;

Node(int data) {//constructor

this->data = data;

next = NULL;

}

};

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

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PHAN HOÀNG NAM - 20215434

# YOUR CODE HERE #\*/

Node \*newNode = new Node(data);//tạo node mới

//gán node mới lên đầu danh sách

newNode->next = head->next;

head->next = newNode;

return head;//trả về head

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

}

//PHAN HOÀNG NAM - 20215434

// print the list content on a line

void print(Node\* head) {

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

# YOUR CODE HERE #\*/

Node \*p = head->next;//khai báo node tạm p = node tiếp theo của head

while(p!=NULL){//lặp tới cuối danh sách

printf("%d ", p->data);//in ra giá trị node p

p=p->next;//gán p = node tiếp theo

}

printf("\n");//xuống dòng

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

}

//PHAN HOÀNG NAM - 20215434

// return the new head of the reversed list

Node\* reverse(Node\* h) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PHAN HOÀNG NAM - 20215434

# YOUR CODE HERE #\*/

if (h->next == NULL || h->next->next == NULL) {

return h;//nếu danh sách rỗng thì trả về head

}

Node\* prev = NULL;//khai báo node tạm prev là node trước đó

Node\* current = h->next;//khai báo node tạm current là node hiện tại

Node\* nextNode = NULL;//khai báo node tạm nextNode là node tiếp theo

//PHAN HOÀNG NAM - 20215434

while (current != NULL) {//lặp tới cuối danh sách

nextNode = current->next;//gán giá trị cho node tiếp theo

current->next = prev;//gán giá trị mới cho node tiếp theo sau khi đảo danh sách (là node trước)

prev = current;//gán node hiện tại vào node trước

current = nextNode;//gán node sau và node hiện tại

}

h->next = prev;//trỏ node tiếp theo của head tới đầu danh sách mới

return h;//trả về con trỏ head

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

}//PHAN HOÀNG NAM - 20215434

int main() {

int n, u;

cin >> n;

Node\* head = new Node(1e9);//tạo node head

for (int i = 0; i < n; ++i){//nhập danh sách

cin >> u;//nhập data

head = prepend(head, u);//thêm vào đầu danh sách

}

cout << "Original list: ";

print(head);//in

head = reverse(head);//đảo danh sách

//PHAN HOÀNG NAM - 20215434

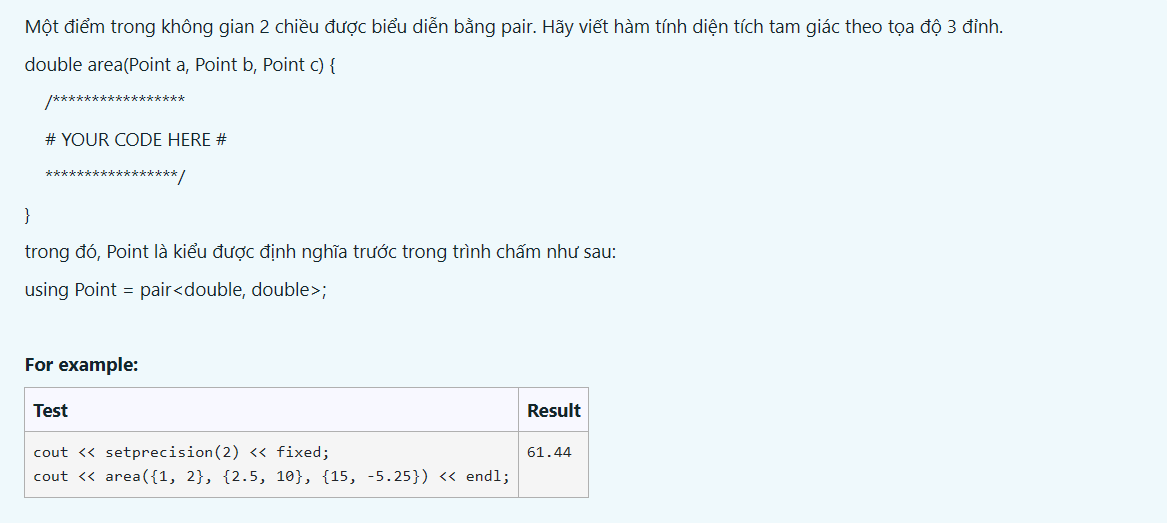
cout << "Reversed list: ";

print(head);//in

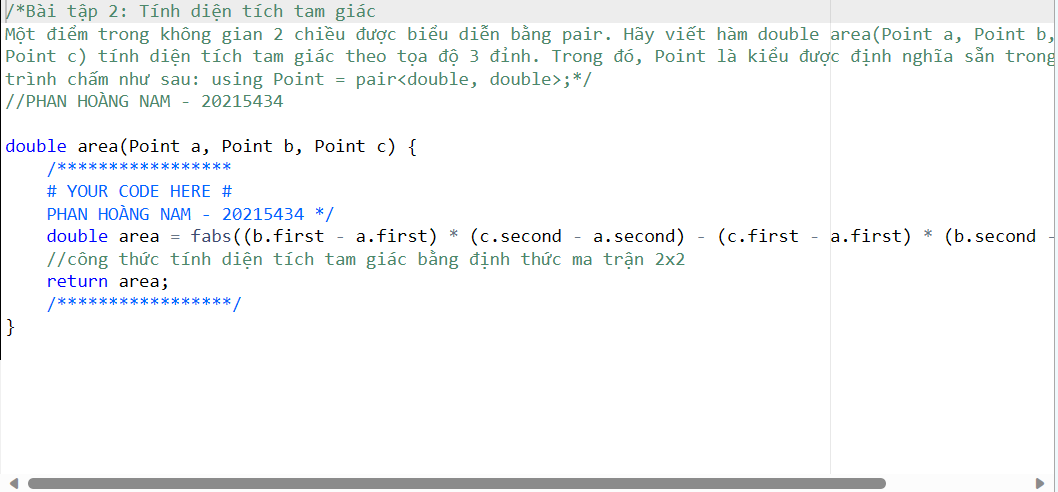
return 0;

}

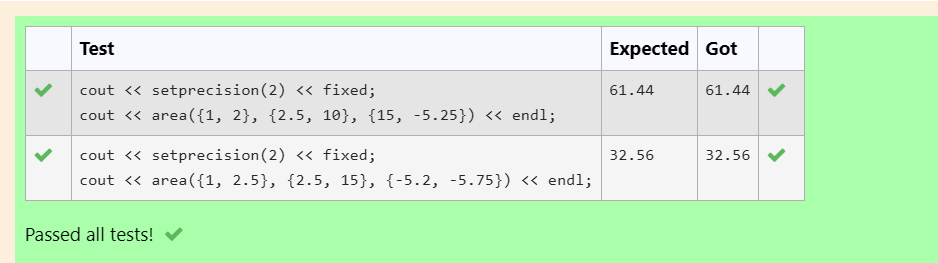
## **Bài 4.2:** Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm tính diện tích tam giác theo tọa độ 3 đỉnh.



Hình 8. Code bài 4.2 ảnh 1



Hình 9. Code bài 4.2 ảnh 2



Hình 10. Test bài 4.2

/\*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>;\*/

//PHAN HOÀNG NAM - 20215434

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

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

# YOUR CODE HERE #

PHAN HOÀNG NAM - 20215434 \*/

double area = fabs((b.first - a.first) \* (c.second - a.second) - (c.first - a.first) \* (b.second - a.second)) / 2;

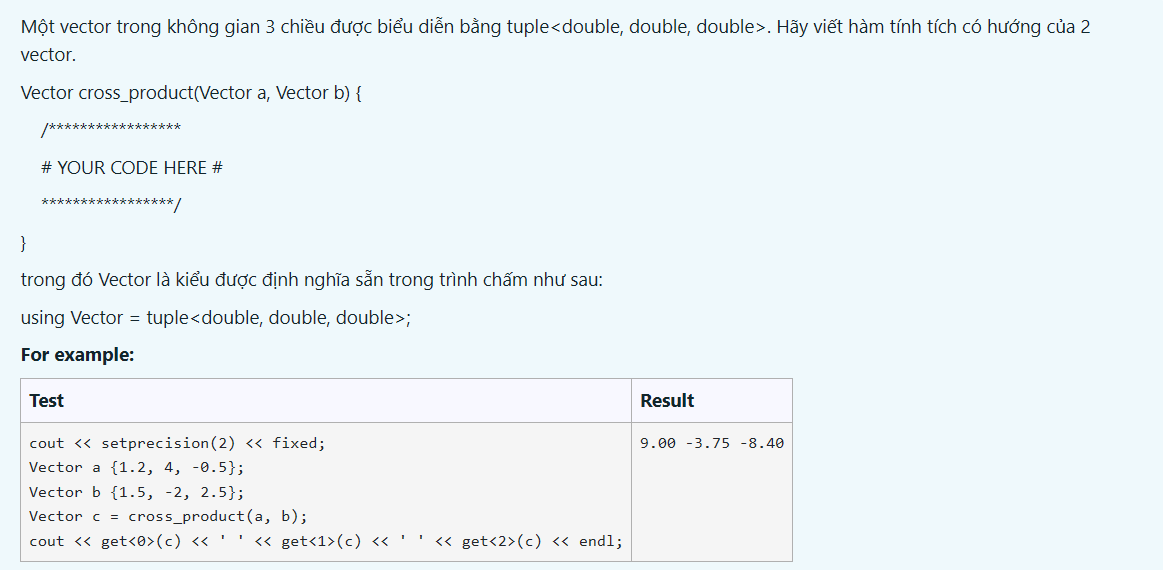
//công thức tính diện tích tam giác bằng định thức ma trận 2x2

return area;

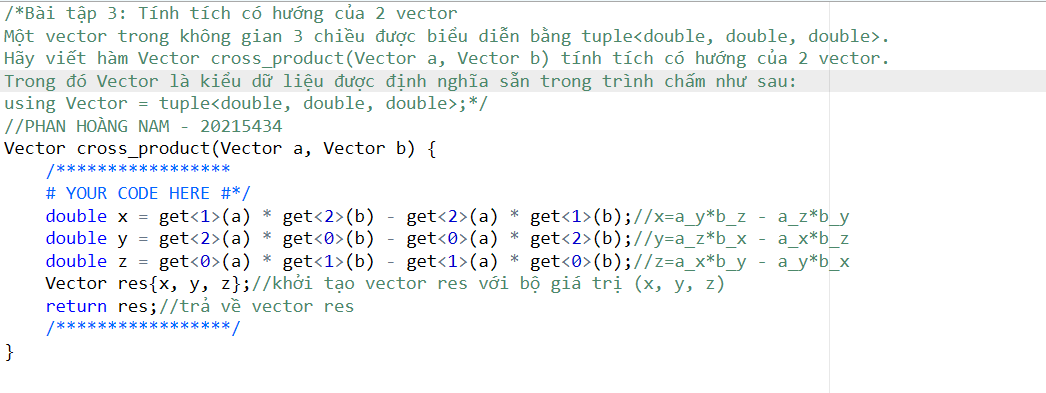
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

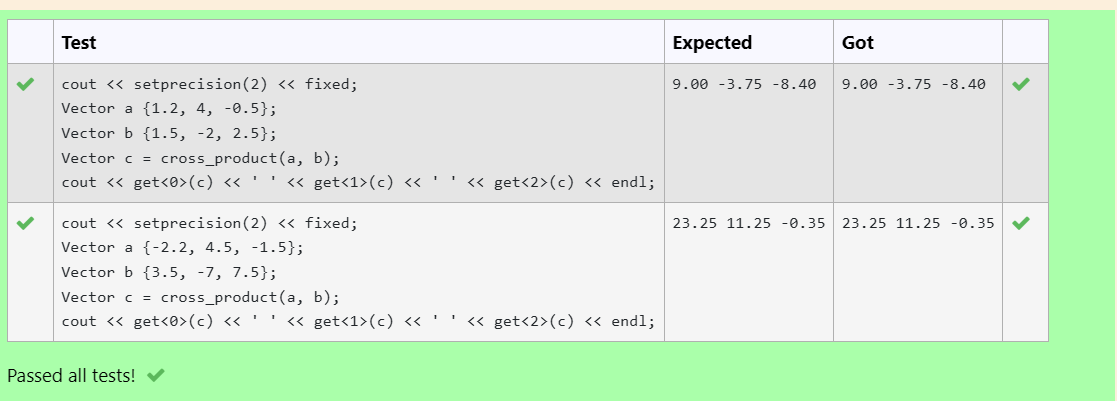
## Bài tập 3: Tính tích có hướng của 2 vector Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product(Vector a, Vector b) tính tích có hướng của 2 vector.



Hình 11. Code bài 4.3 ảnh 1



Hình 12. Code bài 4.3 ảnh 2



Hình 13. Test bài 4.3

/\*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>;\*/

//PHAN HOÀNG NAM - 20215434

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

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

# YOUR CODE HERE #\*/

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

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

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

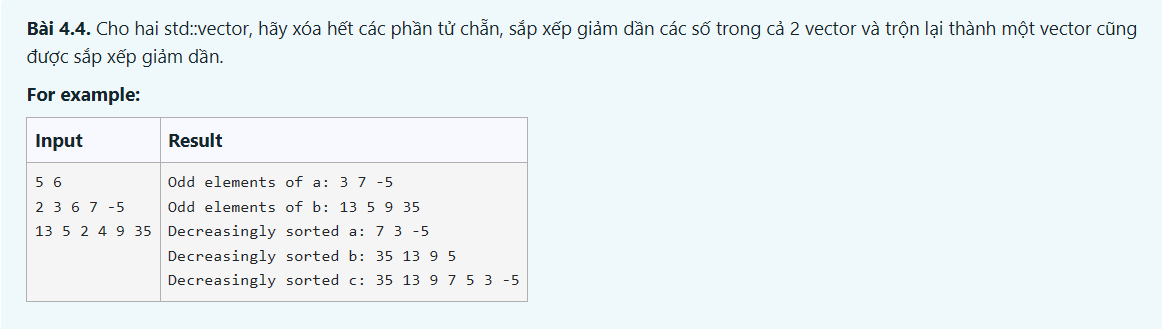
Vector res{x, y, z};//khởi tạo vector res với bộ giá trị (x, y, z)

return res;//trả về vector res

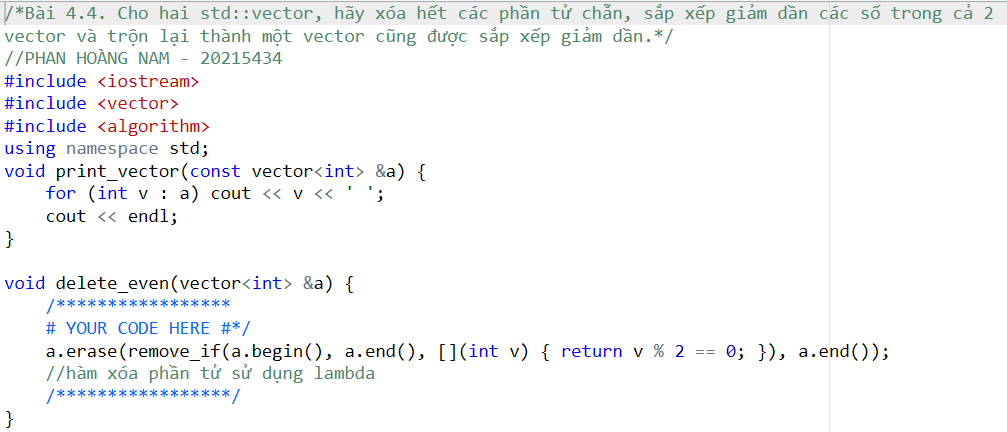
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

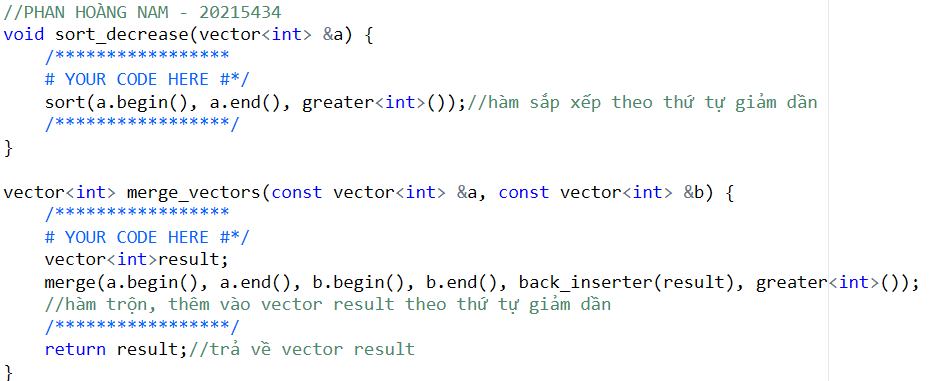
## Bài 4.4. Cho hai std::vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần.



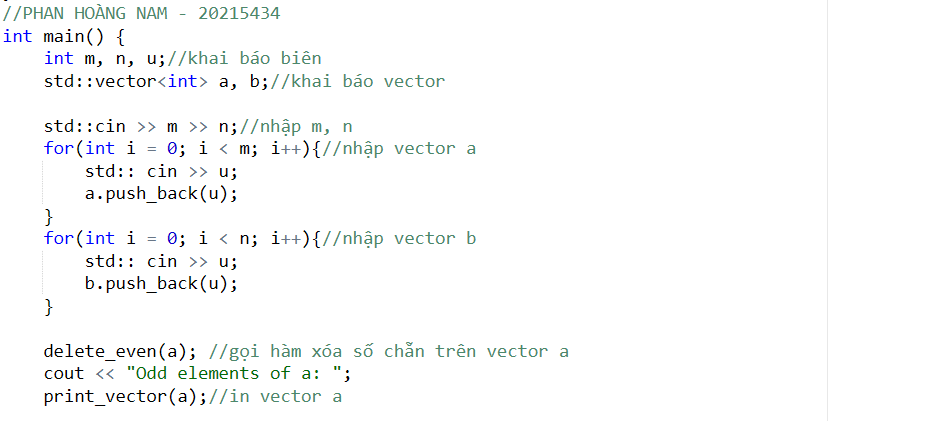
Hình 14. Code bài 4.4 ảnh 1



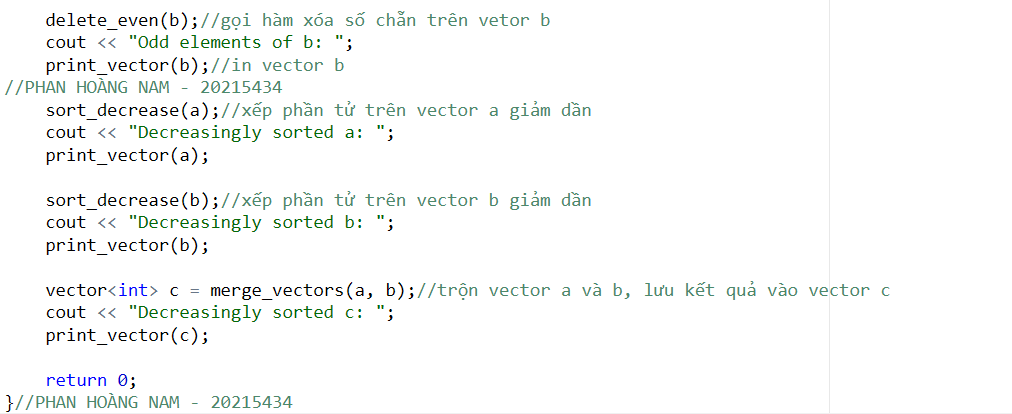
Hình 15. Code bài 4.4 ảnh 2



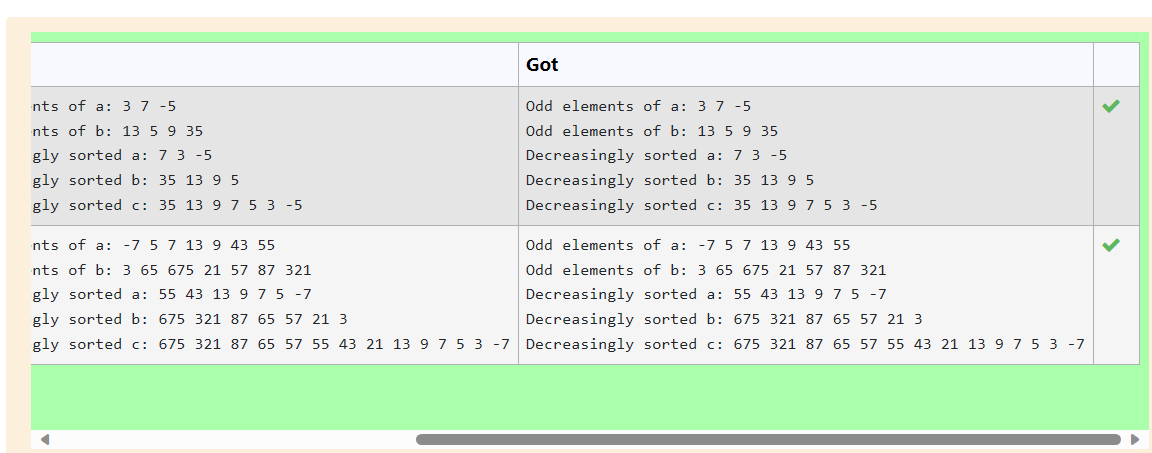
Hình 16. Code bài 4.4 ảnh 3



Hình 17. Code bài 4.4 ảnh 4



Hình 18. Code bài 4.4 ảnh 5



Hình 19. Test bài 4.4

/\*Bài 4.4. Cho hai std::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.\*/

//PHAN HOÀNG NAM - 20215434

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

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

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

cout << endl;

}

void delete\_even(vector<int> &a) {

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

# YOUR CODE HERE #\*/

a.erase(remove\_if(a.begin(), a.end(), [](int v) { return v % 2 == 0; }), a.end());

//hàm xóa phần tử sử dụng lambda

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

}

//PHAN HOÀNG NAM - 20215434

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

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

# YOUR CODE HERE #\*/

sort(a.begin(), a.end(), greater<int>());//hàm sắp xếp theo thứ tự giảm dần

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

}

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

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

# YOUR CODE HERE #\*/

vector<int>result;

merge(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(result), greater<int>());

//hàm trộn, thêm vào vector result theo thứ tự giảm dần

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

return result;//trả về vector result

}

//PHAN HOÀNG NAM - 20215434

int main() {

int m, n, u;//khai báo biên

std::vector<int> a, b;//khai báo vector

std::cin >> m >> n;//nhập m, n

for(int i = 0; i < m; i++){//nhập vector a

std:: cin >> u;

a.push\_back(u);

}

for(int i = 0; i < n; i++){//nhập vector b

std:: cin >> u;

b.push\_back(u);

}

delete\_even(a); //gọi hàm xóa số chẵn trên vector a

cout << "Odd elements of a: ";

print\_vector(a);//in vector a

delete\_even(b);//gọi hàm xóa số chẵn trên vetor b

cout << "Odd elements of b: ";

print\_vector(b);//in vector b

//PHAN HOÀNG NAM - 20215434

sort\_decrease(a);//xếp phần tử trên vector a giảm dần

cout << "Decreasingly sorted a: ";

print\_vector(a);

sort\_decrease(b);//xếp phần tử trên vector b giảm dần

cout << "Decreasingly sorted b: ";

print\_vector(b);

vector<int> c = merge\_vectors(a, b);//trộn vector a và b, lưu kết quả vào vector c

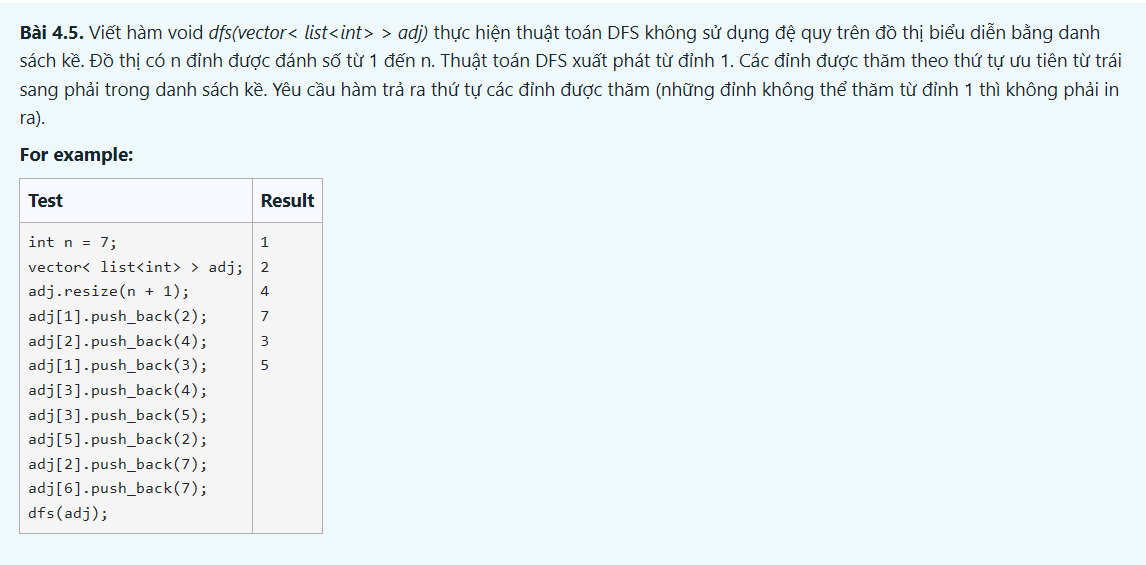
cout << "Decreasingly sorted c: ";

print\_vector(c);

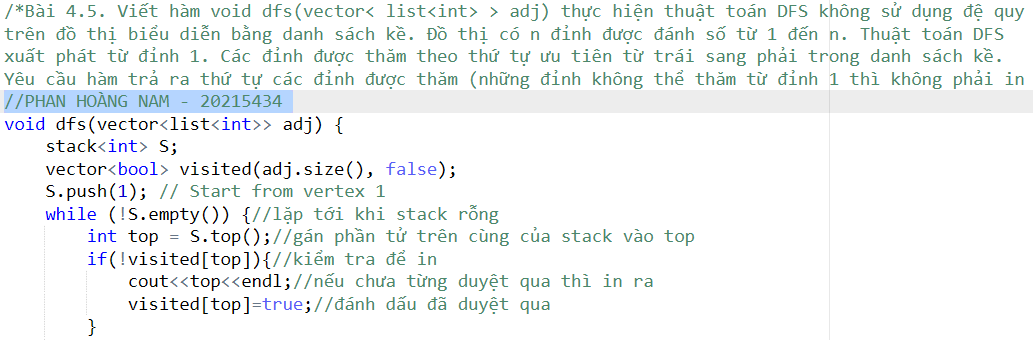
return 0;

}//PHAN HOÀNG NAM - 20215434

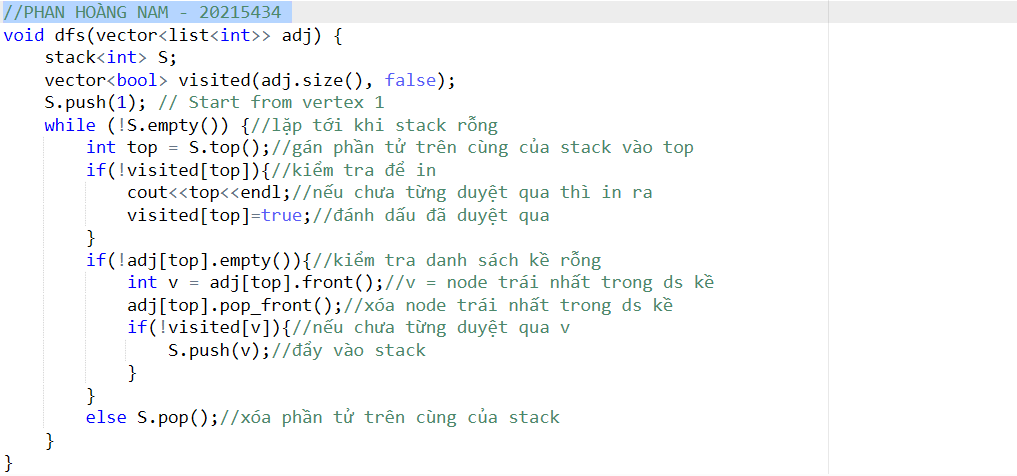
## Bài 4.5. Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).



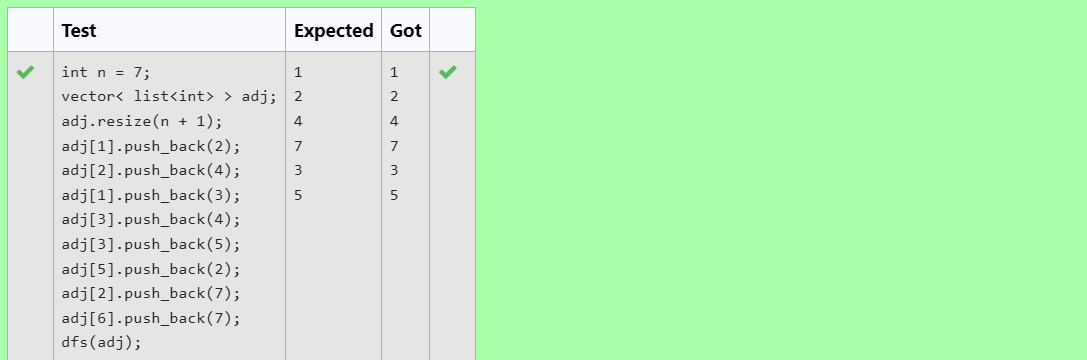
Hình 20. Code bài 4.5 ảnh 1



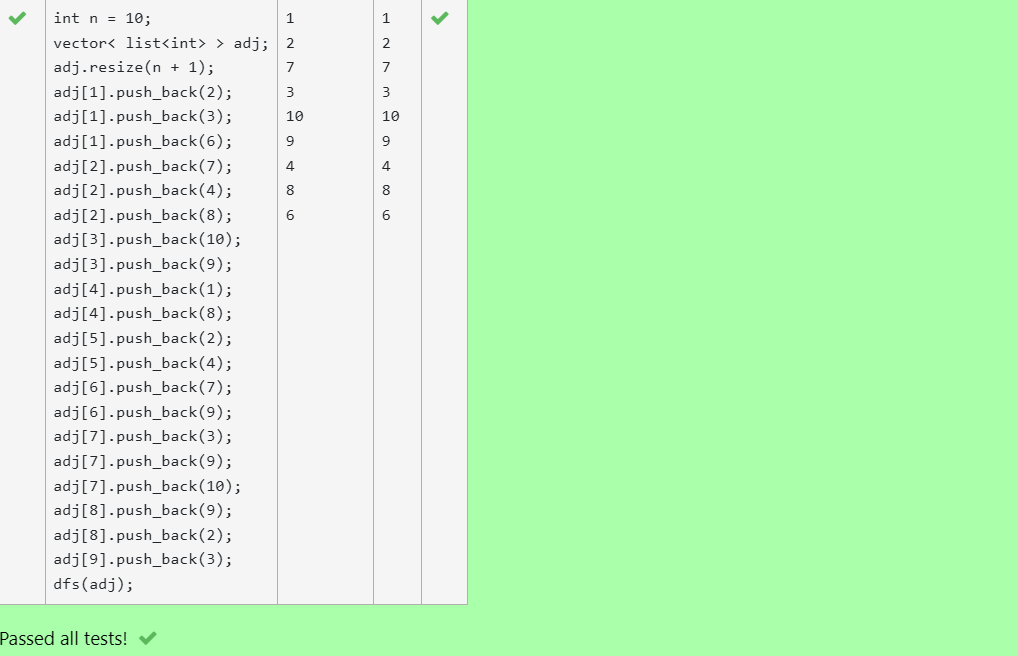
Hình 21. Code bài 4.5 ảnh 2



Hình 22. Code bài 4.5 ảnh 3



Hình 23. Test bài 4.5 ảnh 1



Hình 24. Test bài 4.5 ảnh 2

/\*Bài 4.5. Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy

trên đồ thị biểu diễn bằng danh sách kề. 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).\*/

//PHAN HOÀNG NAM - 20215434

void dfs(vector<list<int>> adj) {

stack<int> S;

vector<bool> visited(adj.size(), false);

S.push(1); // Start from vertex 1

while (!S.empty()) {//lặp tới khi stack rỗng

int top = S.top();//gán phần tử trên cùng của stack vào top

if(!visited[top]){//kiểm tra để in

cout<<top<<endl;//nếu chưa từng duyệt qua thì in ra

visited[top]=true;//đánh dấu đã duyệt qua

}

if(!adj[top].empty()){//kiểm tra danh sách kề rỗng

int v = adj[top].front();//v = node trái nhất trong ds kề

adj[top].pop\_front();//xóa node trái nhất trong ds kề

if(!visited[v]){//nếu chưa từng duyệt qua v

S.push(v);//đẩy vào stack

}

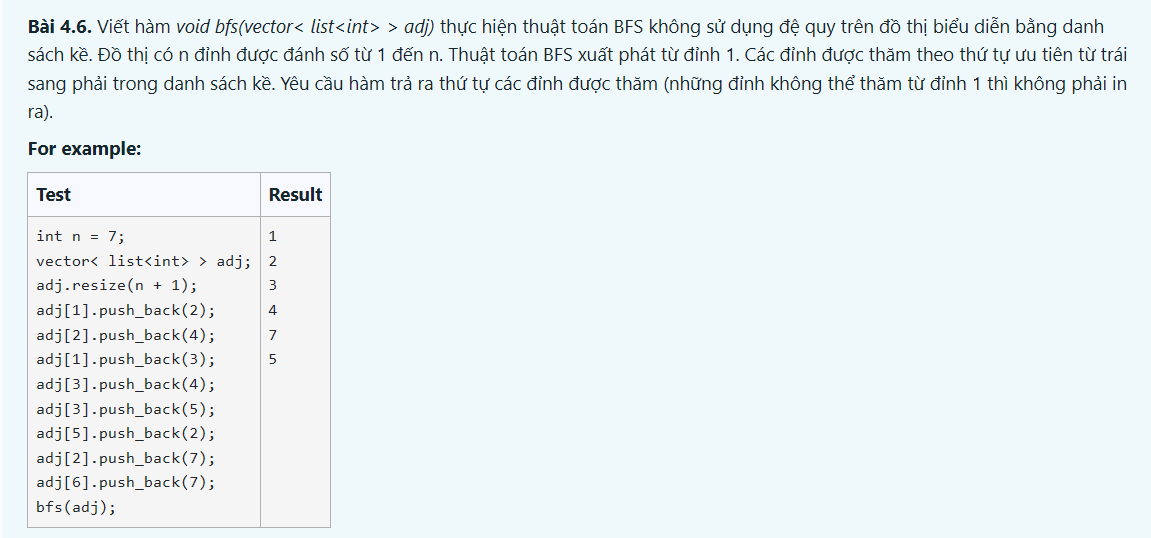
}

else S.pop();//xóa phần tử trên cùng của stack

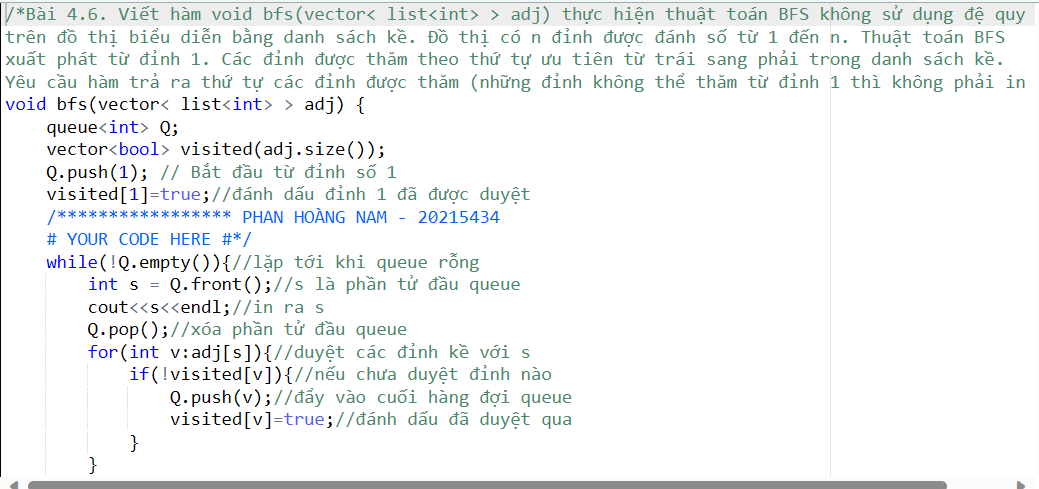
}

}

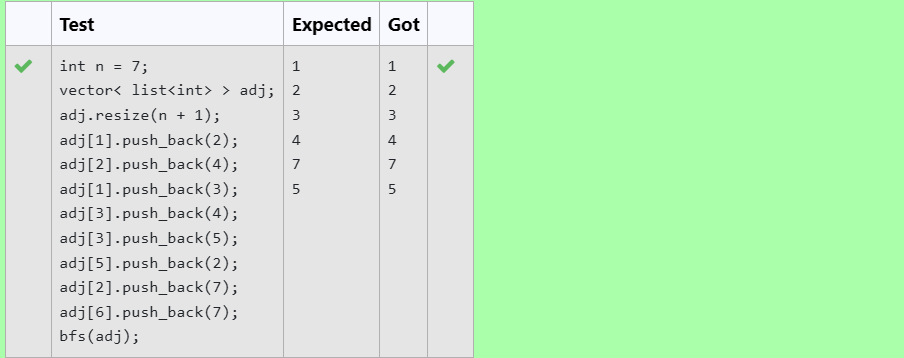
## Bài 4.6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).



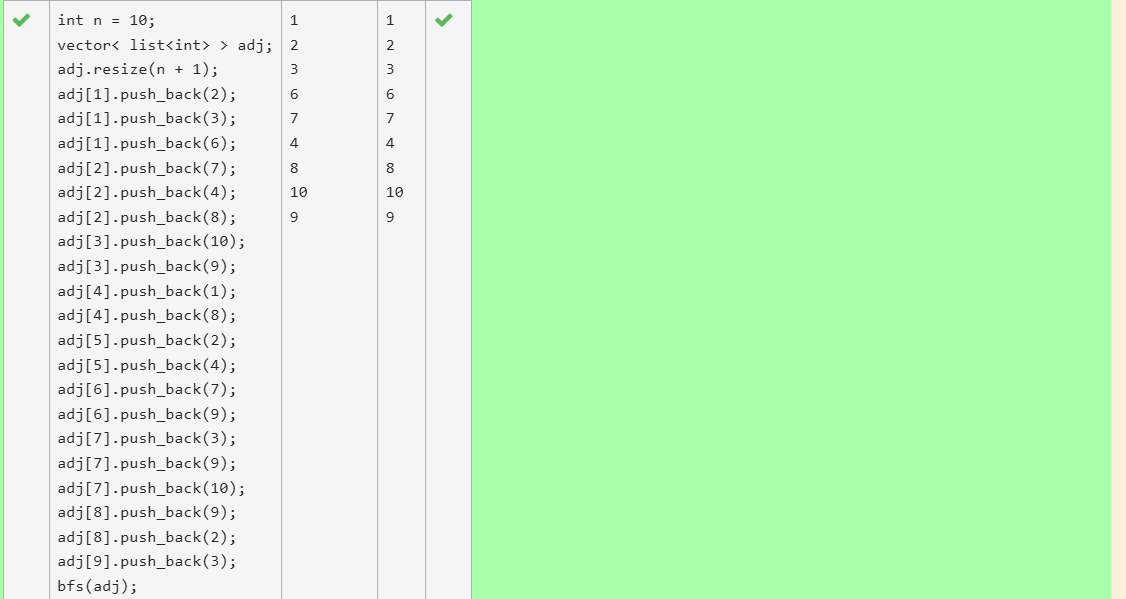
Hình 25. Code bào 4.6 ảnh 1



Hình 26. Code bào 4.6 ảnh 2



Hình 27. Test bài 4.6 ảnh 1



Hình 28. Test bài 4.6 ảnh 2

/\*Bài 4.6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy

trên đồ thị biểu diễn bằng danh sách kề. 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).\*/

void bfs(vector< list<int> > adj) {

queue<int> Q;

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

Q.push(1); // Bắt đầu từ đỉnh số 1

visited[1]=true;//đánh dấu đỉnh 1 đã được duyệt

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PHAN HOÀNG NAM - 20215434

# YOUR CODE HERE #\*/

while(!Q.empty()){//lặp tới khi queue rỗng

int s = Q.front();//s là phần tử đầu queue

cout<<s<<endl;//in ra s

Q.pop();//xóa phần tử đầu queue

for(int v:adj[s]){//duyệt các đỉnh kề với s

if(!visited[v]){//nếu chưa duyệt đỉnh nào

Q.push(v);//đẩy vào cuối hàng đợi queue

visited[v]=true;//đánh dấu đã duyệt qua

}

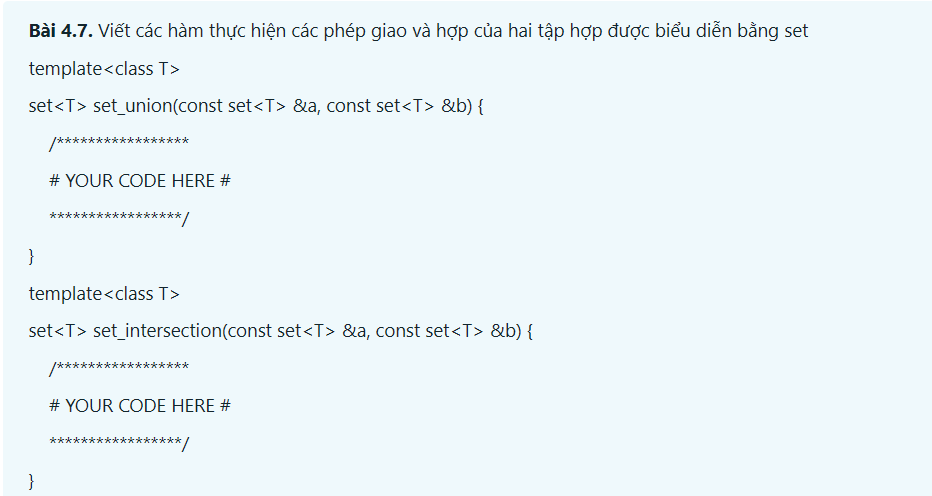
}

}

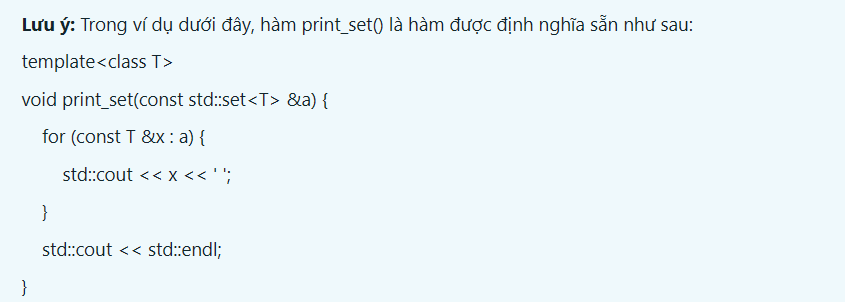
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

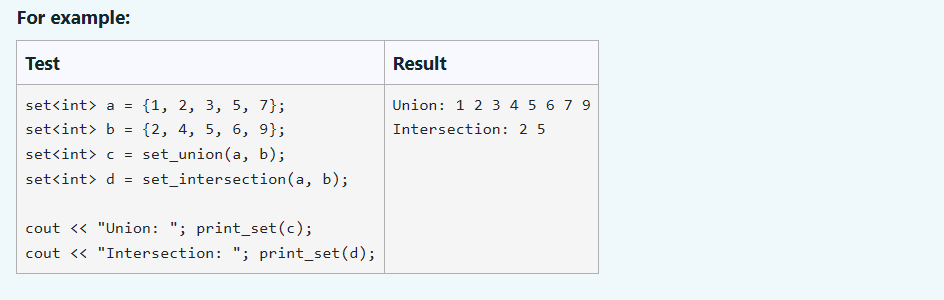
## Bài 4.7. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set



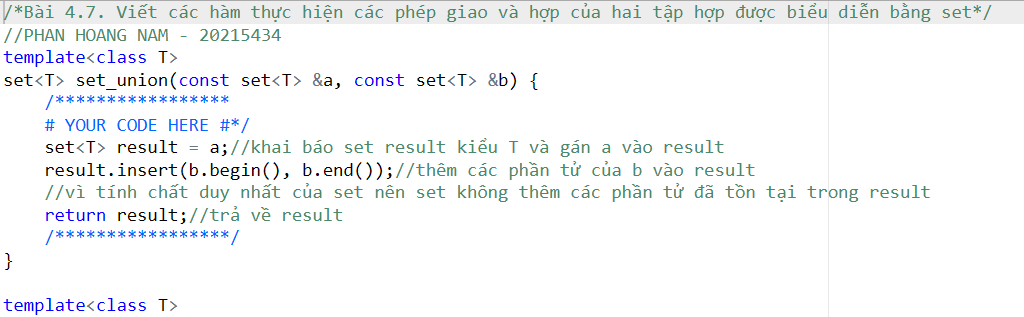
Hình 29. Code bài 4.7 ảnh 1



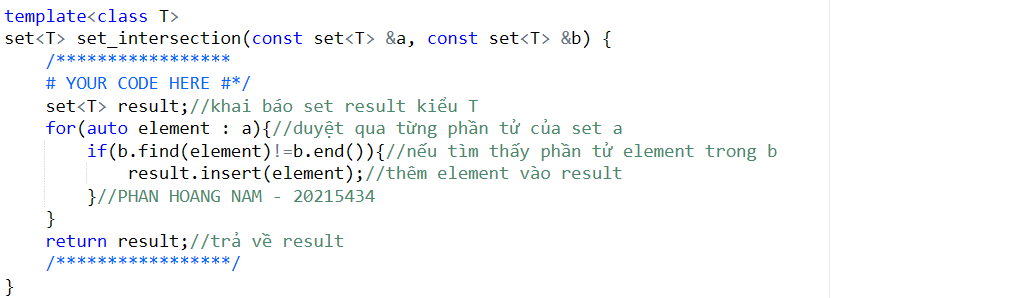
Hình 30. Code bài 4.7 ảnh 2



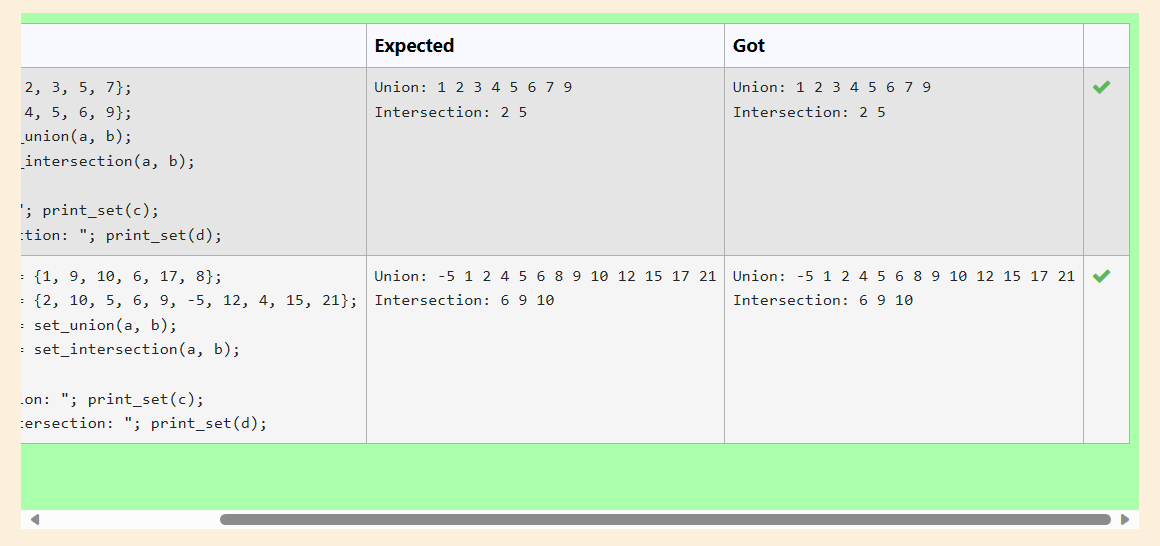
Hình 31. Code bài 4.7 ảnh 3



Hình 32. Code bài 4.7 ảnh 4



Hình 33. Code bài 4.7 ảnh 5



Hình 34. Test bài 4.7

/\*Bài 4.7. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set\*/

//PHAN HOANG NAM - 20215434

template<class T>

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

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

# YOUR CODE HERE #\*/

set<T> result = a;//khai báo set result kiểu T và gán a vào result

result.insert(b.begin(), b.end());//thêm các phần tử của b vào result

//vì tính chất duy nhất của set nên set không thêm các phần tử đã tồn tại trong result

return result;//trả về result

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

}

template<class T>

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

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

# YOUR CODE HERE #\*/

set<T> result;//khai báo set result kiểu T

for(auto element : a){//duyệt qua từng phần tử của set a

if(b.find(element)!=b.end()){//nếu tìm thấy phần tử element trong b

result.insert(element);//thêm element vào result

}//PHAN HOANG NAM - 20215434

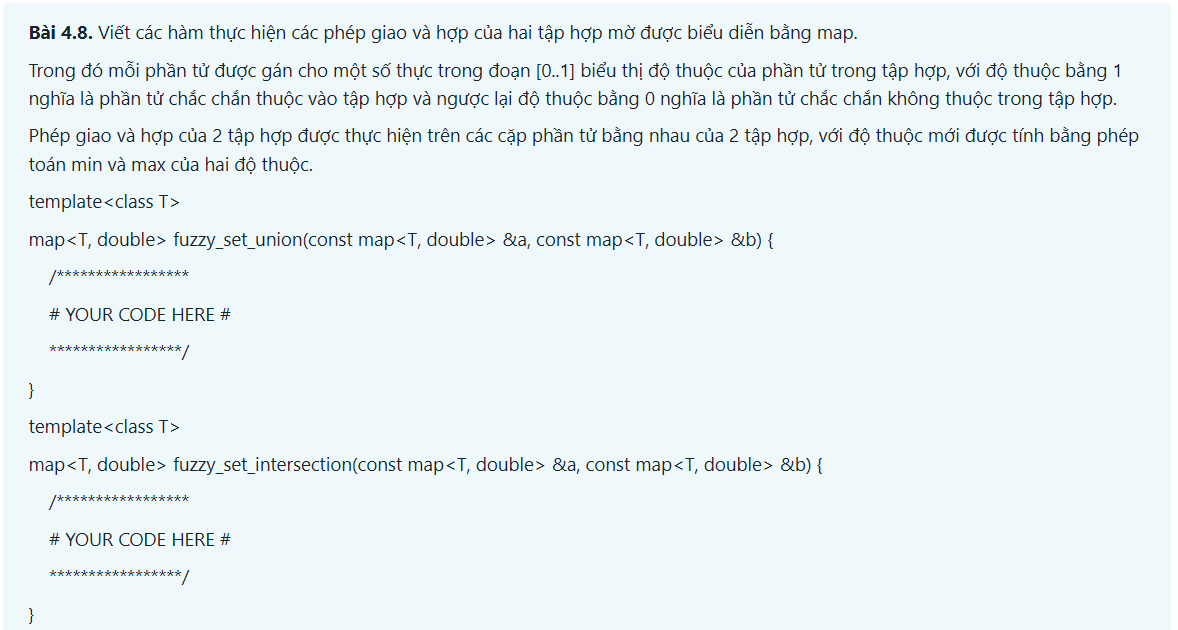
}

return result;//trả về result

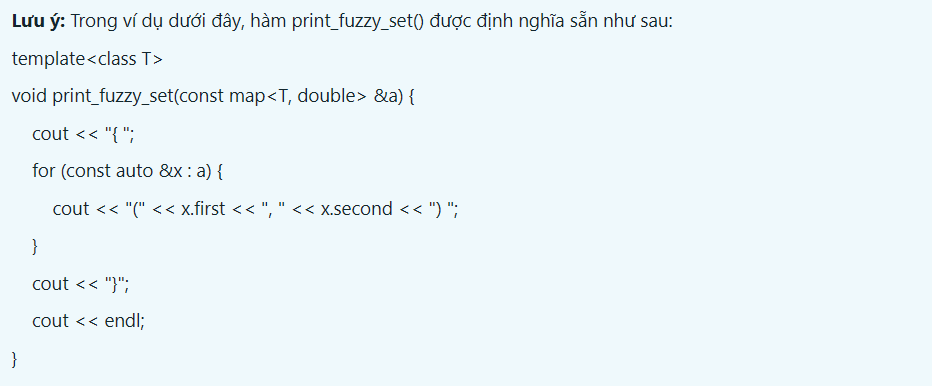
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

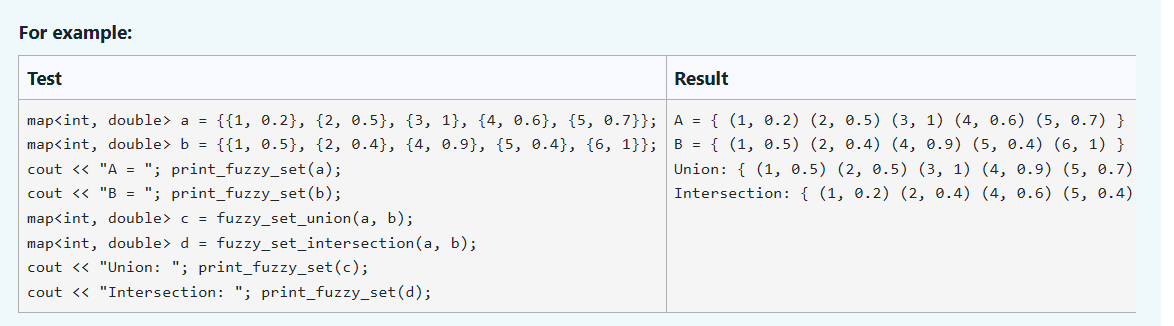
## Bài 4.8. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. Trong đó mỗi phần tử được gán cho một số thực trong đoạn [0..1] biểu thị độ thuộc của phần tử trong tập hợp, với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 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.



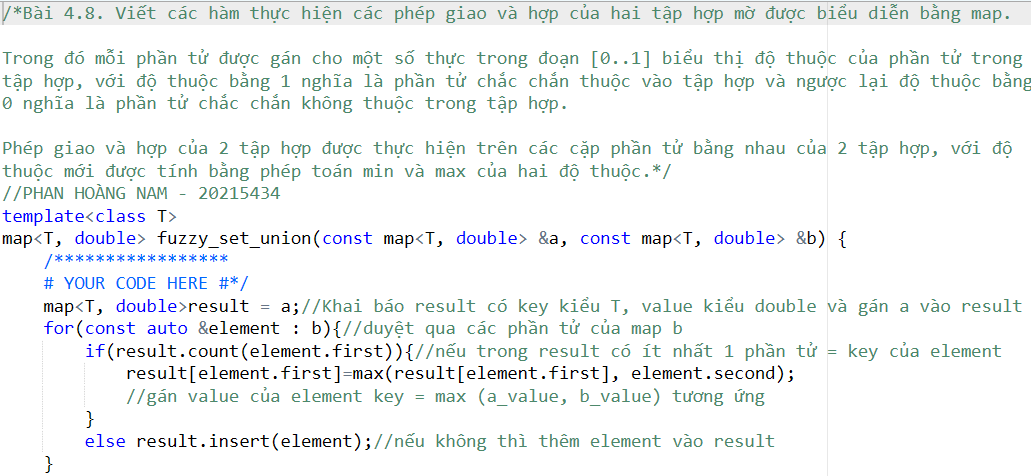
Hình 35. Code bài 4.8 ảnh 1



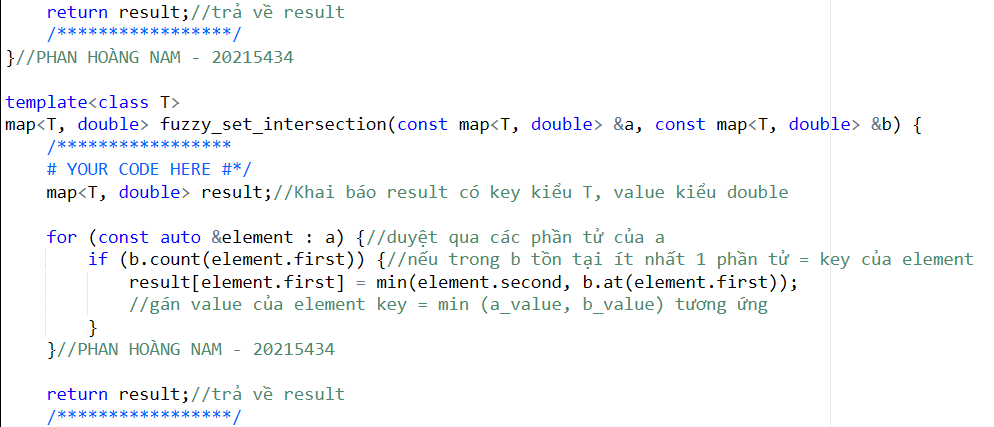
Hình 36. Code bài 4.8 ảnh 2



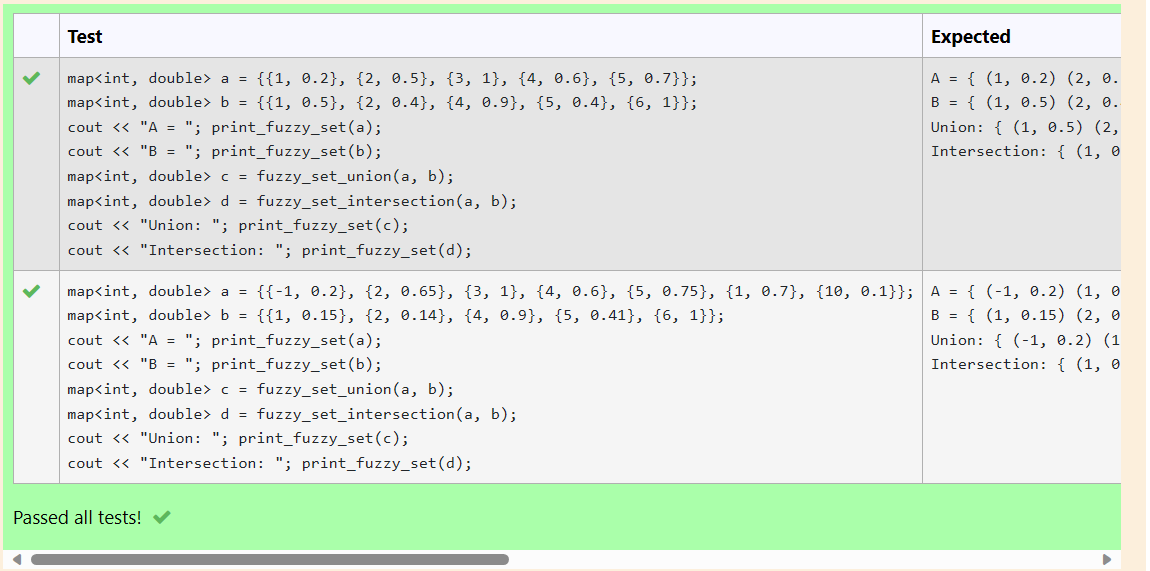
Hình 37. Code bài 4.8 ảnh 3



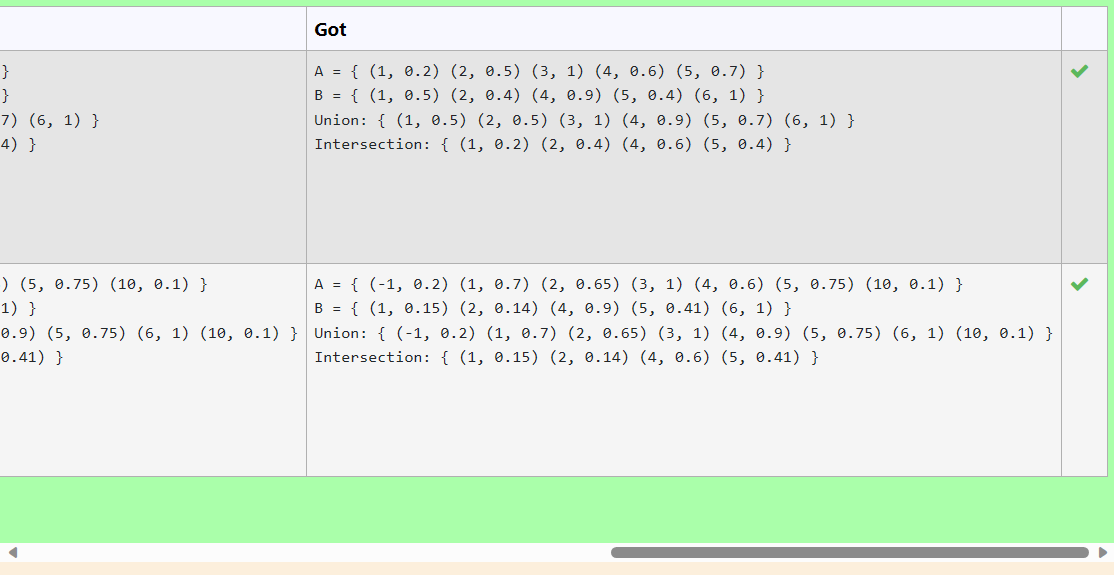
Hình 38. Code bài 4.8 ảnh 4



Hình 39. Code bài 4.8 ảnh 5

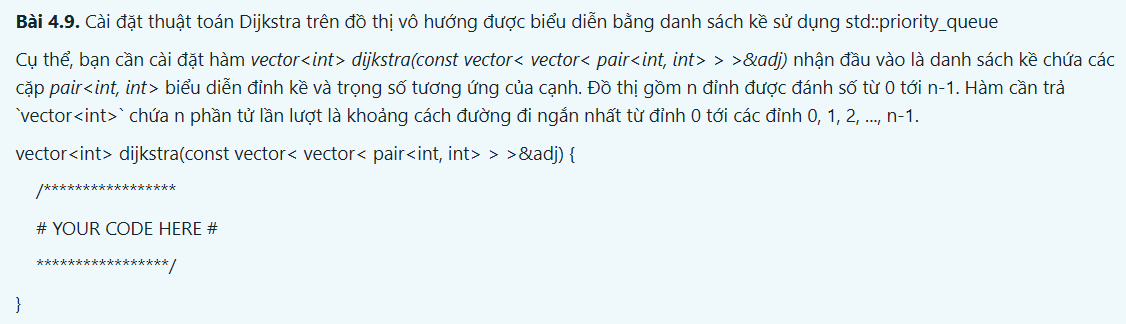


Hình 40. Test bài 4.8 ảnh 1

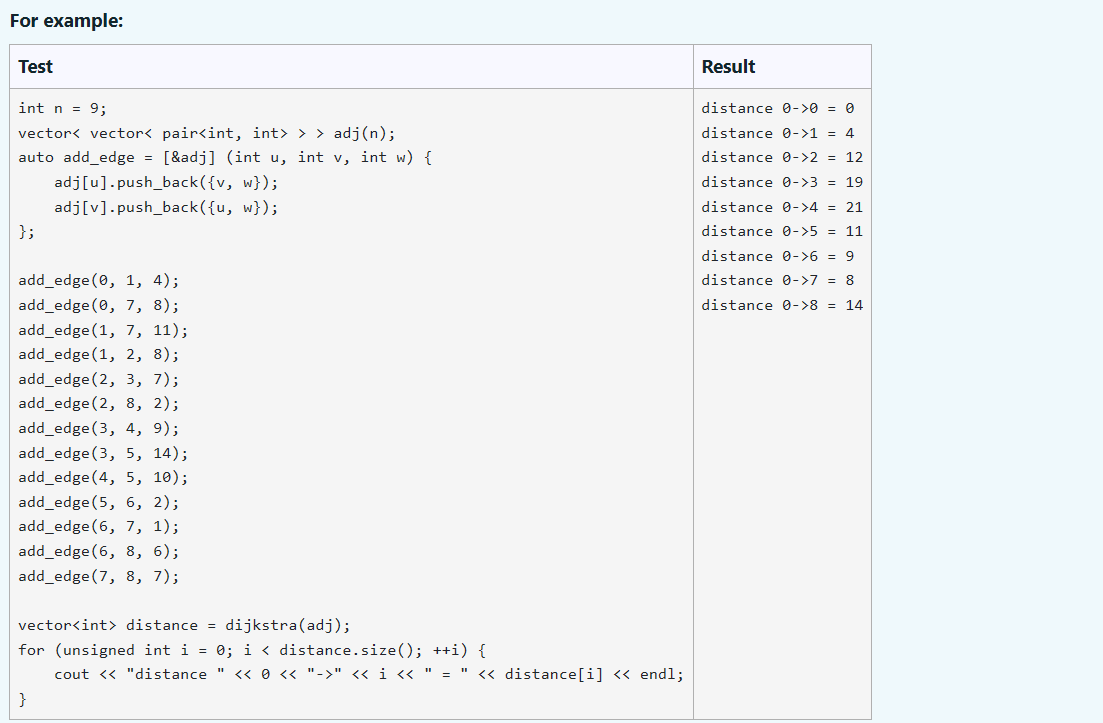


Hình 41. Test bài 4.8 ảnh 2

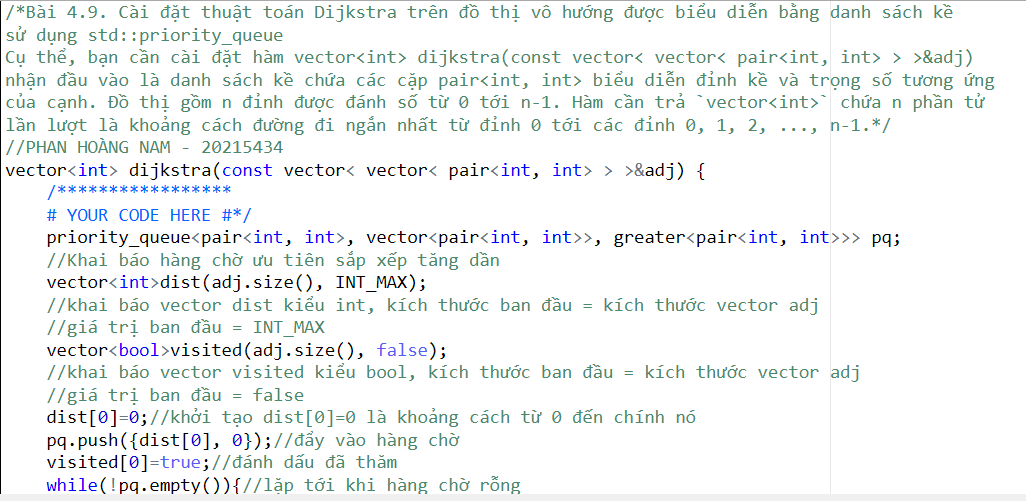
## Bài 4.9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority\_queue Cụ thể, bạn cần cài đặt hàm vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) nhận đầu vào là danh sách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng 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.



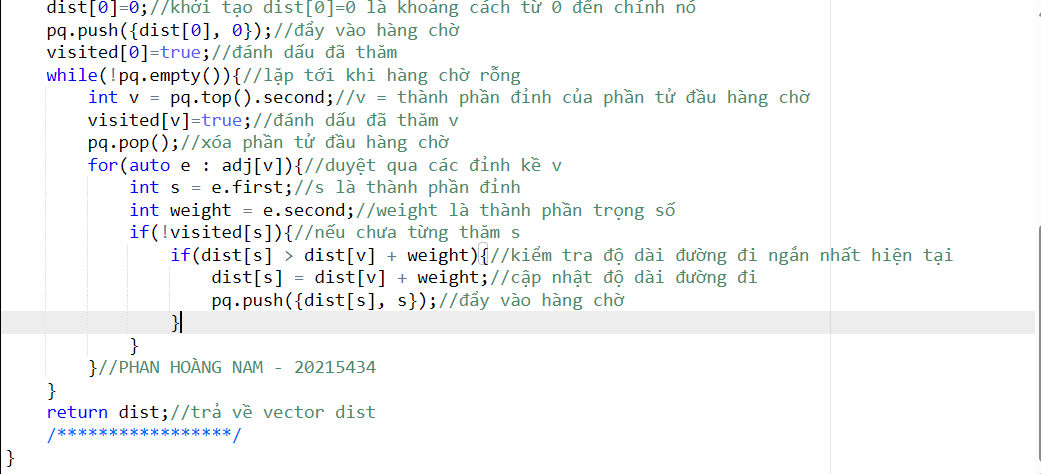
Hình 42. Code bài 4.9 ảnh 1



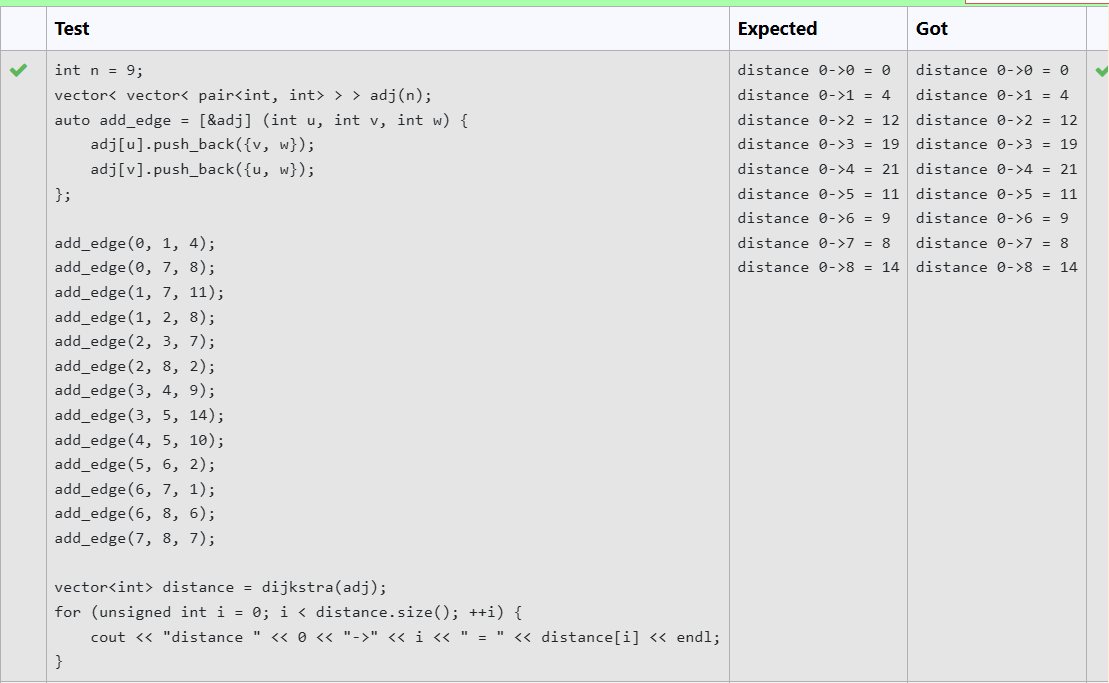
Hình 43. Code bài 4.9 ảnh 2



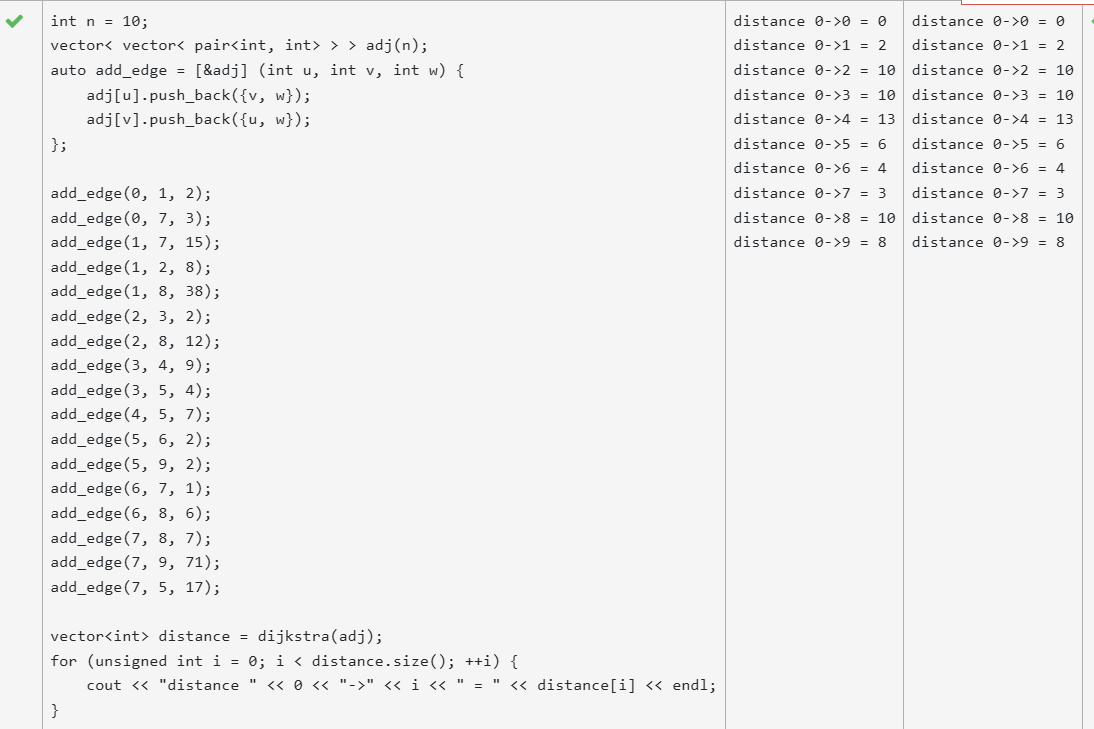
Hình 44. Code bài 4.9 ảnh 3



Hình 45. Code bài 4.9 ảnh 4



Hình 46. Test bài 4.9 ảnh 1



Hình 47. Test bài 4.9 ảnh 2

/\*Bài 4.9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề

sử dụng std::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.\*/

//PHAN HOÀNG NAM - 20215434

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

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

# YOUR CODE HERE #\*/

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

//Khai báo hàng chờ ưu tiên sắp xếp tăng dần

vector<int>dist(adj.size(), INT\_MAX);

//khai báo vector dist kiểu int, kích thước ban đầu = kích thước vector adj

//giá trị ban đầu = INT\_MAX

vector<bool>visited(adj.size(), false);

//khai báo vector visited kiểu bool, kích thước ban đầu = kích thước vector adj

//giá trị ban đầu = false

dist[0]=0;//khởi tạo dist[0]=0 là khoảng cách từ 0 đến chính nó

pq.push({dist[0], 0});//đẩy vào hàng chờ

visited[0]=true;//đánh dấu đã thăm

while(!pq.empty()){//lặp tới khi hàng chờ rỗng

int v = pq.top().second;//v = thành phần đỉnh của phần tử đầu hàng chờ

visited[v]=true;//đánh dấu đã thăm v

pq.pop();//xóa phần tử đầu hàng chờ

for(auto e : adj[v]){//duyệt qua các đỉnh kề v

int s = e.first;//s là thành phần đỉnh

int weight = e.second;//weight là thành phần trọng số

if(!visited[s]){//nếu chưa từng thăm s

if(dist[s] > dist[v] + weight){//kiểm tra độ dài đường đi ngắn nhất hiện tại

dist[s] = dist[v] + weight;//cập nhật độ dài đường đi

pq.push({dist[s], s});//đẩy vào hàng chờ

}

}

}//PHAN HOÀNG NAM - 20215434

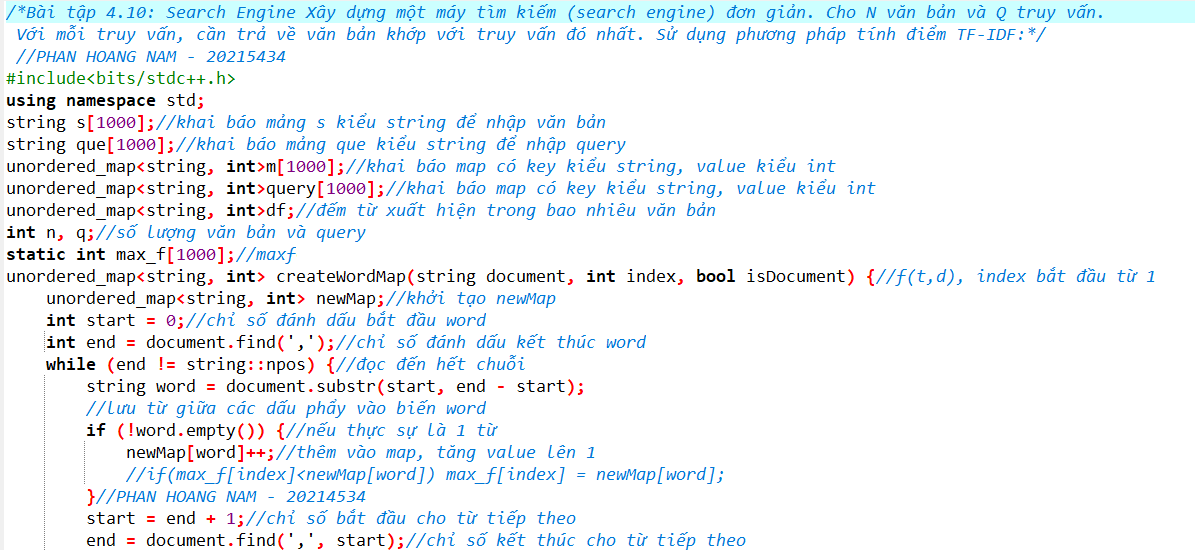
}

return dist;//trả về vector dist

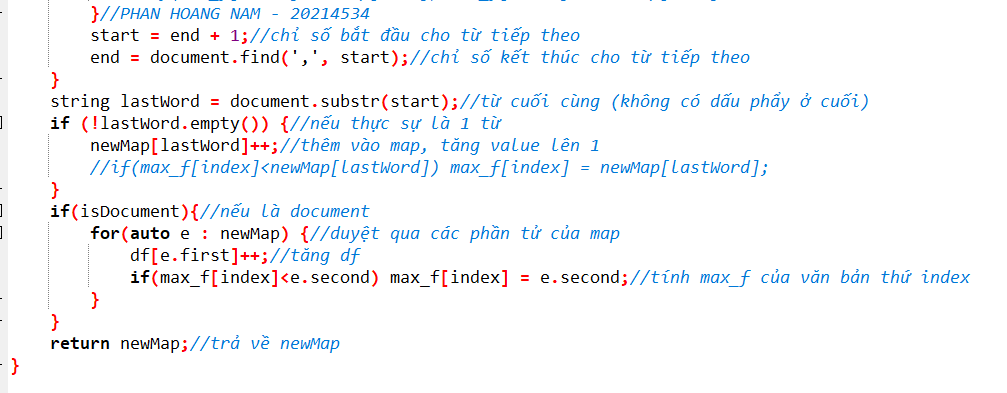
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

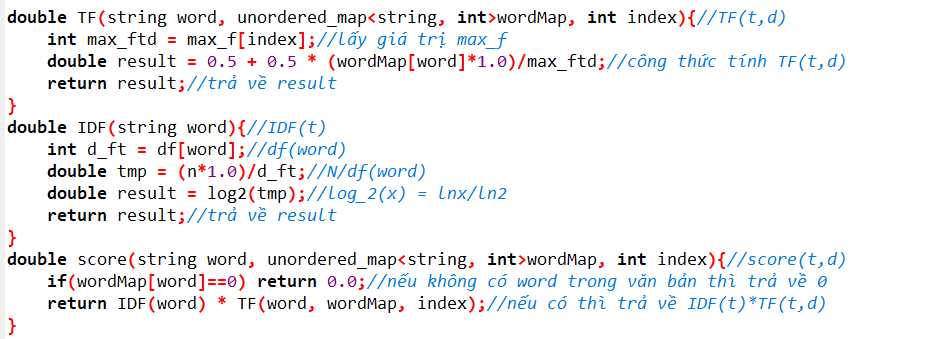
## Bài tập 4.10: Search Engine Xây dựng một máy tìm kiếm (search engine) đơn giản. Cho N văn bản và Q truy vấn. Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất. Sử dụng phương pháp tính điểm TF-IDF:



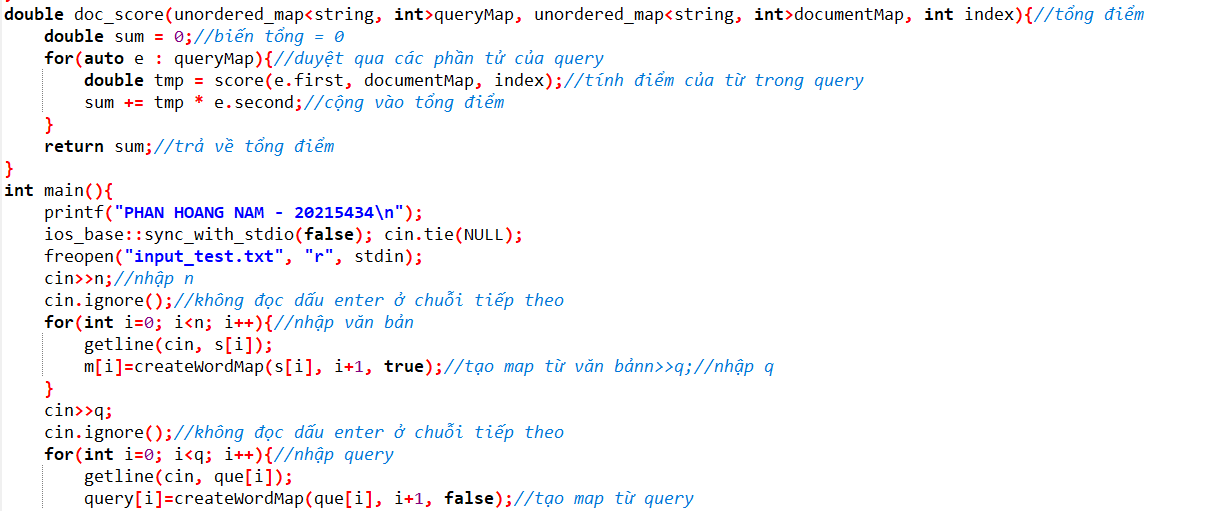
Hình 48. Code bài 4.10 ảnh 1



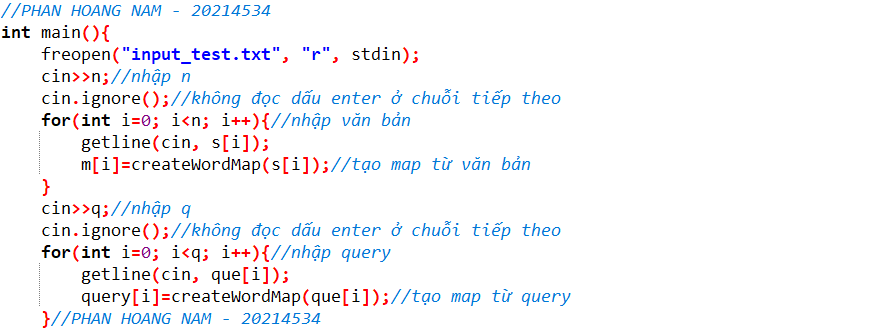
Hình 49. Code bài 4.10 ảnh 2



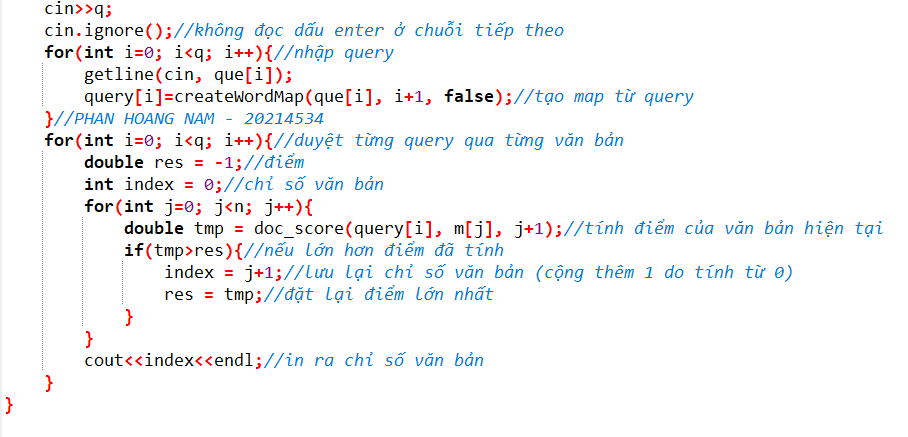
Hình 50. Code bài 4.10 ảnh 3



Hình 51. Code bài 4.10 ảnh 4



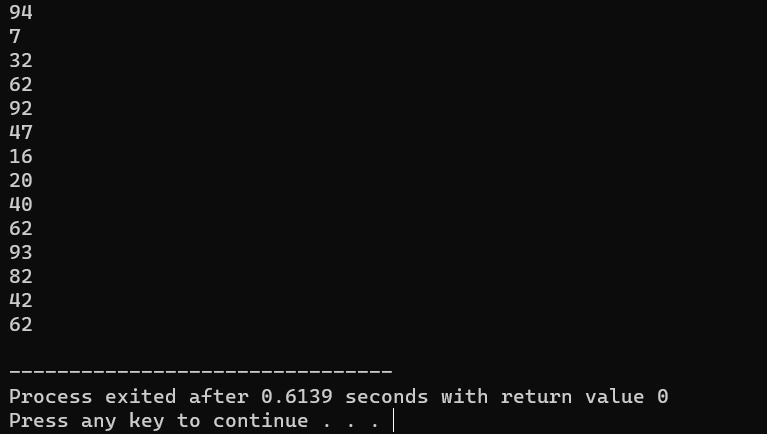
Hình 52. Code bài 4.10 ảnh 5



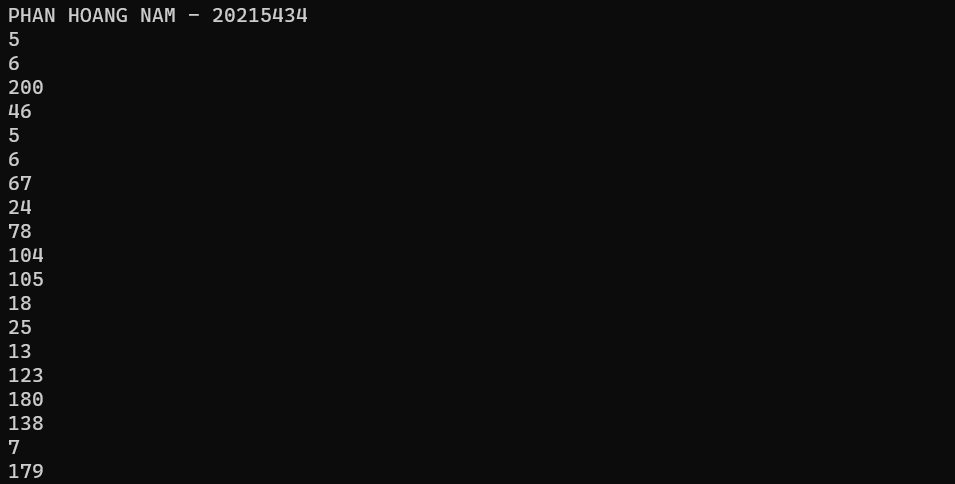
Hình 53. Code bài 4.10 ảnh 6



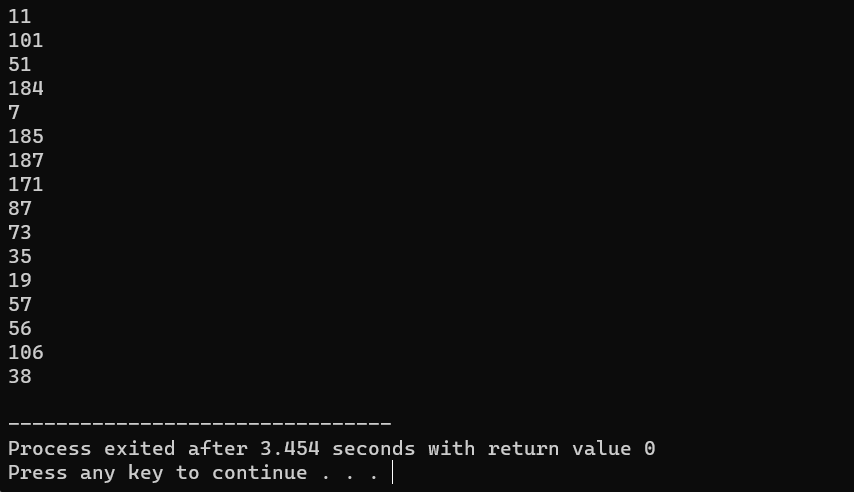
Hình 54. Test case 1 bài 4.10 ảnh 1



Hình 55. Test case 1 bài 4.10 ảnh 2



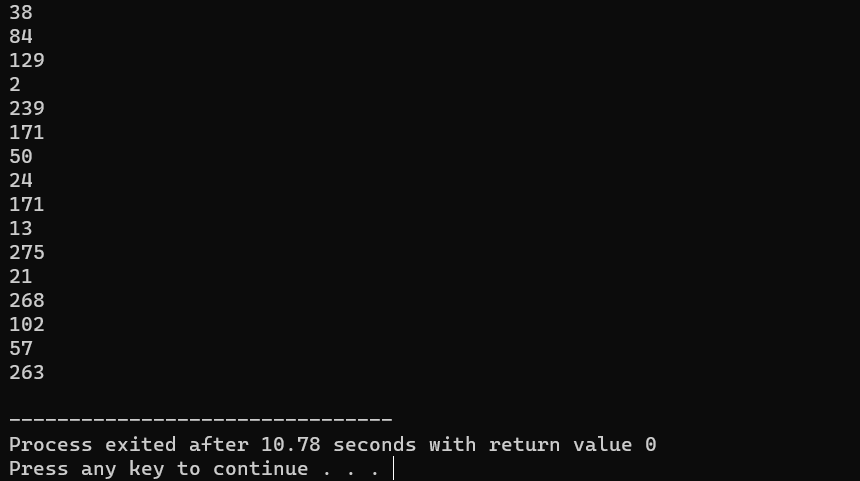
Hình 56. Test case 2 bài 4.10 ảnh 1



Hình 57. Test case 2 bài 4.10 ảnh 2



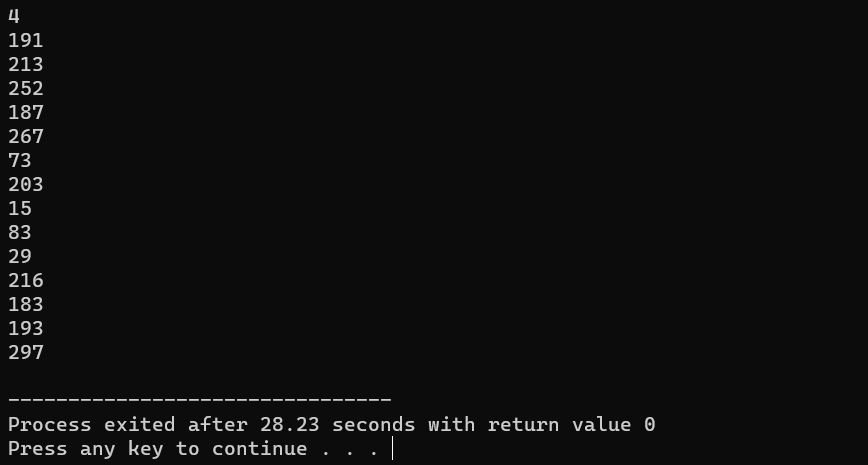
Hình 58. Test case 3 bài 4.10 ảnh 1



Hình 59. Test case 3 bài 4.10 ảnh 2



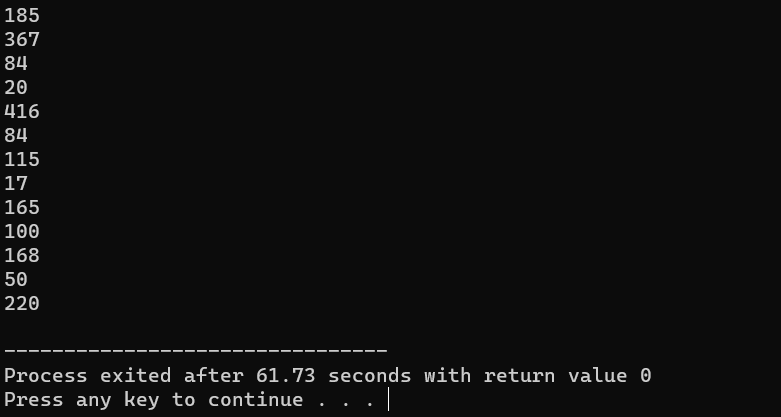
Hình 60. Test case 4 bài 4.10 ảnh 1



Hình 61. Test case 4 bài 4.10 ảnh 2



Hình 62. Test case 5 bài 4.10 ảnh 1



Hình 63. Test case 5 bài 4.10 ảnh 2

/\*Bài tập 4.10: Search Engine Xây dựng một máy tìm kiếm (search engine) đơn giản. Cho N văn bản và Q 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:\*/

//PHAN HOANG NAM - 20215434

#include<bits/stdc++.h>

using namespace std;

string s[1000];//khai báo mảng s kiểu string để nhập văn bản

string que[1000];//khai báo mảng que kiểu string để nhập query

unordered\_map<string, int>m[1000];//khai báo map có key kiểu string, value kiểu int

unordered\_map<string, int>query[1000];//khai báo map có key kiểu string, value kiểu int

unordered\_map<string, int>df;//đếm từ xuất hiện trong bao nhiêu văn bản

int n, q;//số lượng văn bản và query

static int max\_f[1000];//maxf

unordered\_map<string, int> createWordMap(string document, int index, bool isDocument) {//f(t,d), index bắt đầu từ 1

unordered\_map<string, int> newMap;//khởi tạo newMap

int start = 0;//chỉ số đánh dấu bắt đầu word

int end = document.find(',');//chỉ số đánh dấu kết thúc word

while (end != string::npos) {//đọc đến hết chuỗi

string word = document.substr(start, end - start);

//lưu từ giữa các dấu phẩy vào biến word

if (!word.empty()) {//nếu thực sự là 1 từ

newMap[word]++;//thêm vào map, tăng value lên 1

//if(max\_f[index]<newMap[word]) max\_f[index] = newMap[word];

}//PHAN HOANG NAM - 20214534

start = end + 1;//chỉ số bắt đầu cho từ tiếp theo

end = document.find(',', start);//chỉ số kết thúc cho từ tiếp theo

}

string lastWord = document.substr(start);//từ cuối cùng (không có dấu phẩy ở cuối)

if (!lastWord.empty()) {//nếu thực sự là 1 từ

newMap[lastWord]++;//thêm vào map, tăng value lên 1

//if(max\_f[index]<newMap[lastWord]) max\_f[index] = newMap[lastWord];

}

if(isDocument){//nếu là document

for(auto e : newMap) {//duyệt qua các phần tử của map

df[e.first]++;//tăng df

if(max\_f[index]<e.second) max\_f[index] = e.second;//tính max\_f của văn bản thứ index

}

}

return newMap;//trả về newMap

}

double TF(string word, unordered\_map<string, int>wordMap, int index){//TF(t,d)

int max\_ftd = max\_f[index];//lấy giá trị max\_f

double result = 0.5 + 0.5 \* (wordMap[word]\*1.0)/max\_ftd;//công thức tính TF(t,d)

return result;//trả về result

}

double IDF(string word){//IDF(t)

int d\_ft = df[word];//df(word)

double tmp = (n\*1.0)/d\_ft;//N/df(word)

double result = log2(tmp);//log\_2(x) = lnx/ln2

return result;//trả về result

}

double score(string word, unordered\_map<string, int>wordMap, int index){//score(t,d)

if(wordMap[word]==0) return 0.0;//nếu không có word trong văn bản thì trả về 0

return IDF(word) \* TF(word, wordMap, index);//nếu có thì trả về IDF(t)\*TF(t,d)

}

double doc\_score(unordered\_map<string, int>queryMap, unordered\_map<string, int>documentMap, int index){//tổng điểm

double sum = 0;//biến tổng = 0

for(auto e : queryMap){//duyệt qua các phần tử của query

double tmp = score(e.first, documentMap, index);//tính điểm của từ trong query

sum += tmp \* e.second;//cộng vào tổng điểm

}

return sum;//trả về tổng điểm

}

int main(){

printf("PHAN HOANG NAM - 20215434\n");

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

freopen("input\_test.txt", "r", stdin);

cin>>n;//nhập n

cin.ignore();//không đọc dấu enter ở chuỗi tiếp theo

for(int i=0; i<n; i++){//nhập văn bản

getline(cin, s[i]);

m[i]=createWordMap(s[i], i+1, true);//tạo map từ văn bảnn>>q;//nhập q

}

cin>>q;

cin.ignore();//không đọc dấu enter ở chuỗi tiếp theo

for(int i=0; i<q; i++){//nhập query

getline(cin, que[i]);

query[i]=createWordMap(que[i], i+1, false);//tạo map từ query

}//PHAN HOANG NAM - 20214534

for(int i=0; i<q; i++){//duyệt từng query qua từng văn bản

double res = -1;//điểm

int index = 0;//chỉ số văn bản

for(int j=0; j<n; j++){

double tmp = doc\_score(query[i], m[j], j+1);//tính điểm của văn bản hiện tại

if(tmp>res){//nếu lớn hơn điểm đã tính

index = j+1;//lưu lại chỉ số văn bản (cộng thêm 1 do tính từ 0)

res = tmp;//đặt lại điểm lớn nhất

}

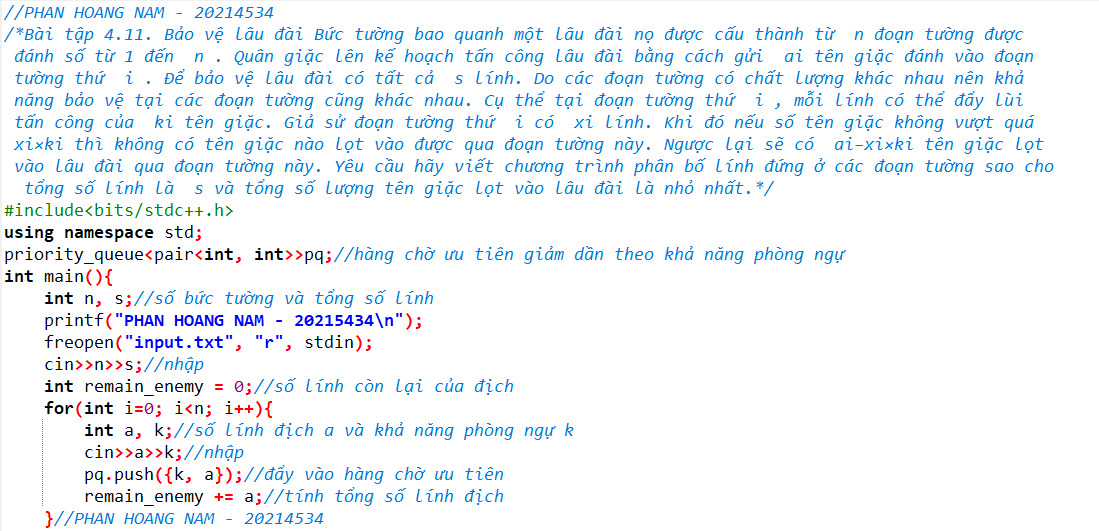
}

cout<<index<<endl;//in ra chỉ số văn bản

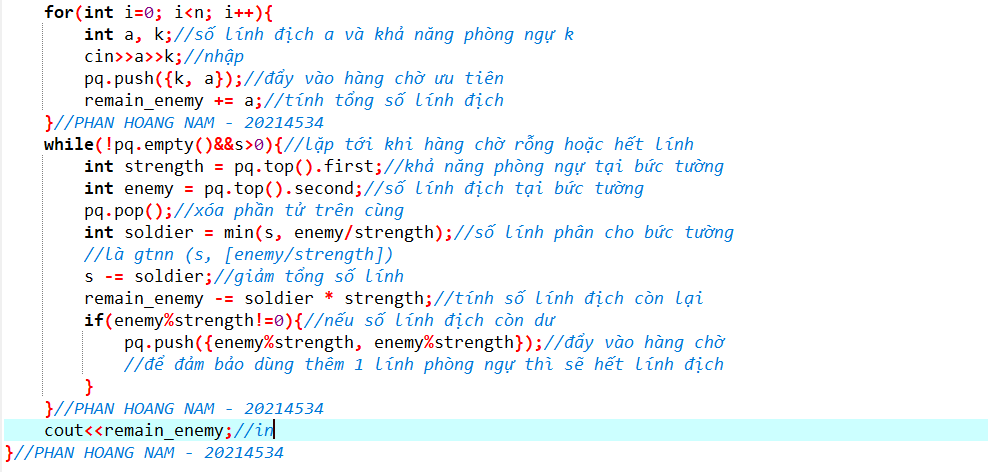
}

}

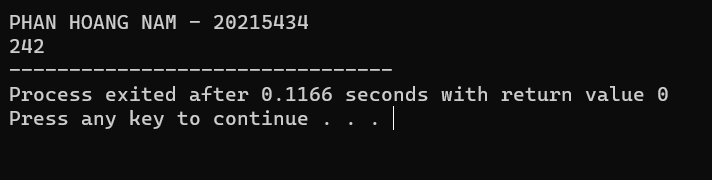
## Bài tập 4.11. Bảo vệ lâu đài Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n . Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi ai tên giặc đánh vào đoạn tường thứ i . Để bảo vệ lâu đài có tất cả s lính. Do các đoạn tường có chất lượng khác nhau nên khả năng bảo vệ tại các đoạn tường cũng khác nhau. Cụ thể tại đoạn tường thứ i , mỗi lính có thể đẩy lùi tấn công của ki tên giặc. Giả sử đoạn tường thứ i có xi lính. Khi đó nếu số tên giặc không vượt quá xi×ki thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai−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 s và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất.



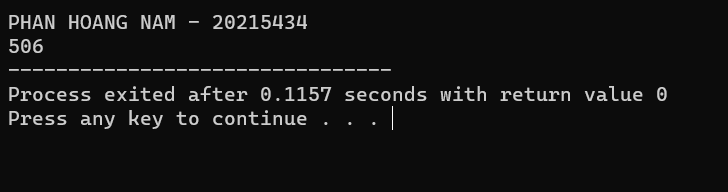
Hình 64. Code bài 4.11 ảnh 1



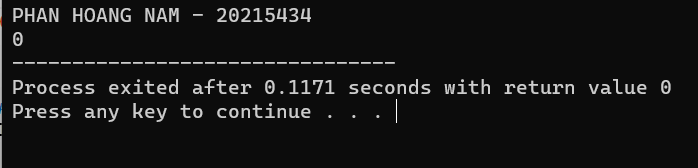
Hình 65. Code bài 4.11 ảnh 2



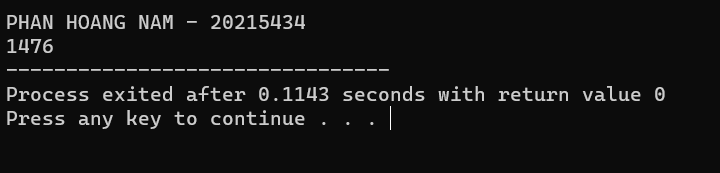
Hình 66. Test case 1 bài 4.11



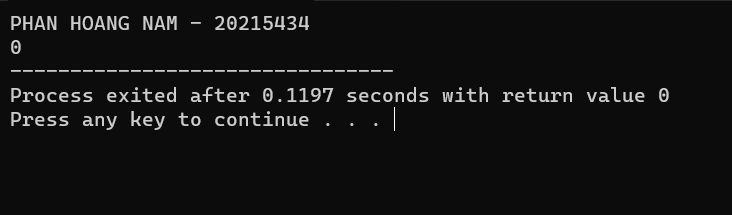
Hình 67. Test case 2 bài 4.11



Hình 68. Test case 3 bài 4.11



Hình 69. Test case 4 bài 4.11



Hình 70. Test case 5 bài 4.11

//PHAN HOANG NAM - 20214534

/\*Bài tập 4.11. Bảo vệ lâu đài Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được

đánh số từ 1 đến n . Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi ai tên giặc đánh vào đoạn

tường thứ i . Để bảo vệ lâu đài có tất cả s 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 i , mỗi lính có thể đẩy lùi

tấn công của ki tên giặc. Giả sử đoạn tường thứ i có xi lính. Khi đó nếu số tên giặc không vượt quá

xi×ki thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai−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 s và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất.\*/

#include<bits/stdc++.h>

using namespace std;

priority\_queue<pair<int, int>>pq;//hàng chờ ưu tiên giảm dần theo khả năng phòng ngự

int main(){

int n, s;//số bức tường và tổng số lính

printf("PHAN HOANG NAM - 20215434\n");

freopen("input.txt", "r", stdin);

cin>>n>>s;//nhập

int remain\_enemy = 0;//số lính còn lại của địch

for(int i=0; i<n; i++){

int a, k;//số lính địch a và khả năng phòng ngự k

cin>>a>>k;//nhập

pq.push({k, a});//đẩy vào hàng chờ ưu tiên

remain\_enemy += a;//tính tổng số lính địch

}//PHAN HOANG NAM - 20214534

while(!pq.empty()&&s>0){//lặp tới khi hàng chờ rỗng hoặc hết lính

int strength = pq.top().first;//khả năng phòng ngự tại bức tường

int enemy = pq.top().second;//số lính địch tại bức tường

pq.pop();//xóa phần tử trên cùng

int soldier = min(s, enemy/strength);//số lính phân cho bức tường

//là gtnn (s, [enemy/strength])

s -= soldier;//giảm tổng số lính

remain\_enemy -= soldier \* strength;//tính số lính địch còn lại

if(enemy%strength!=0){//nếu số lính địch còn dư

pq.push({enemy%strength, enemy%strength});//đẩy vào hàng chờ

//để đảm bảo dùng thêm 1 lính phòng ngự thì sẽ hết lính địch

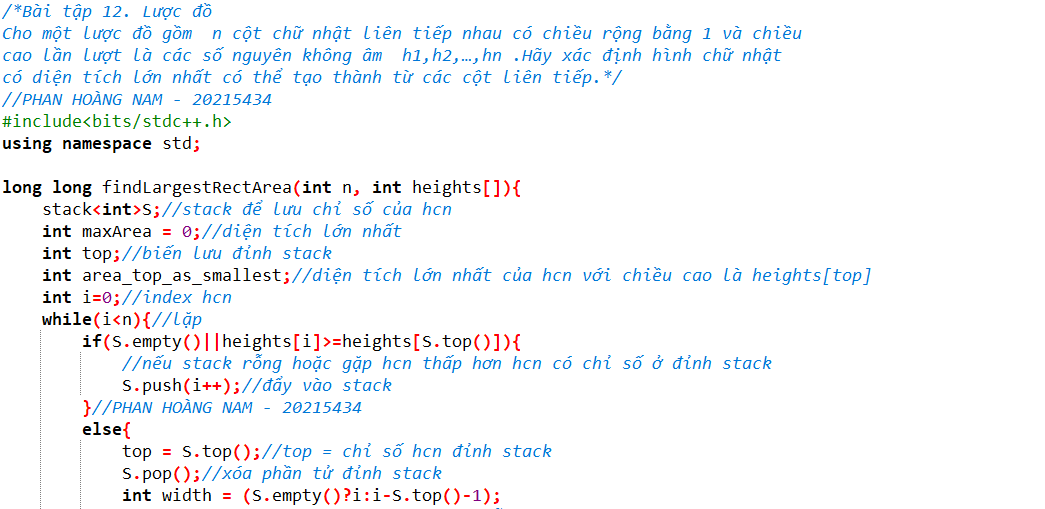
}

}//PHAN HOANG NAM - 20214534

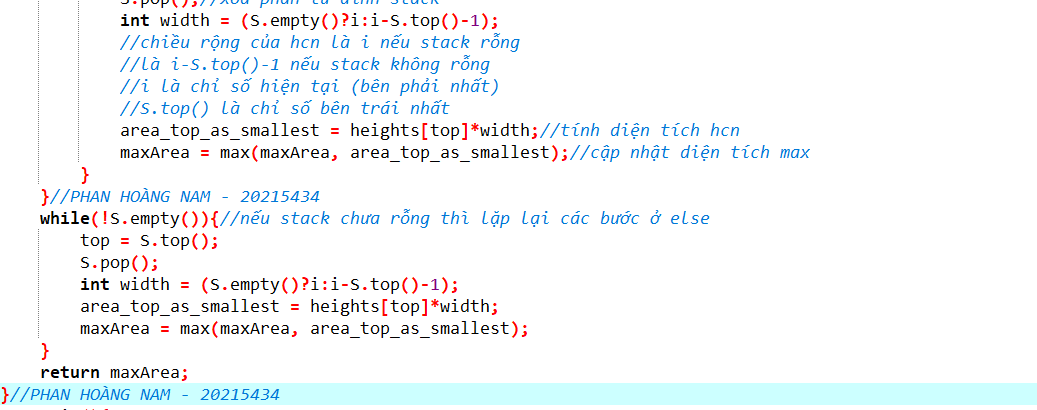
cout<<remain\_enemy;//in

}//PHAN HOANG NAM - 20214534

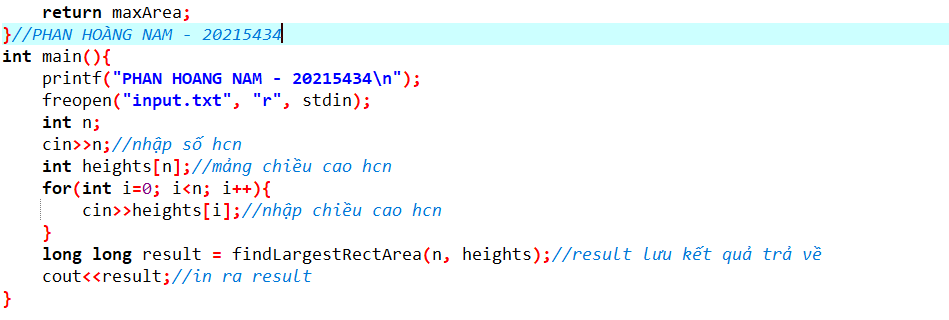
## Bài tập 4.12. Lược đồ Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.



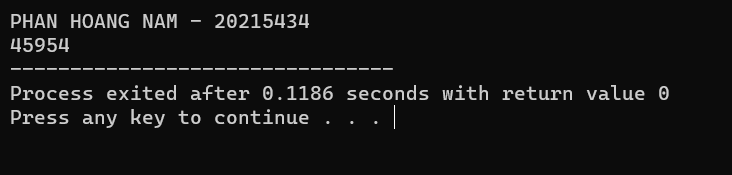
Hình 71. Code bài 4.12 ảnh 1



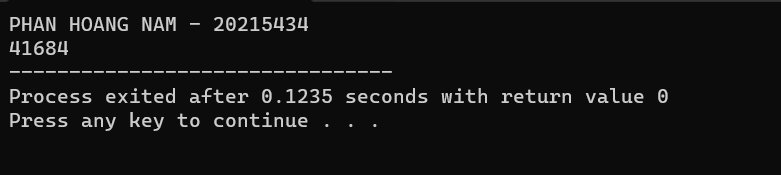
Hình 72. Code bài 4.12 ảnh 2



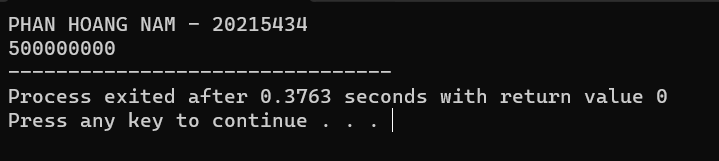
Hình 73. Code bài 4.12 ảnh 3



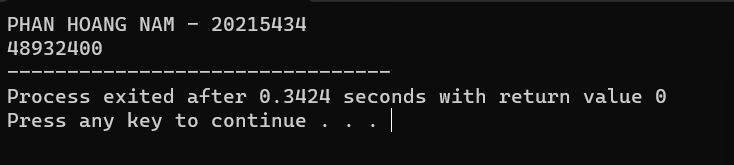
Hình 74. Test case 1 bài 4.12



Hình 75. Test case 2 bài 4.12



Hình 76. Test case 3 bài 4.12



Hình 77. Test case 4 bài 4.12

/\*Bài tập 12. Lược đồ

Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều

cao lần lượt là các số nguyên không âm h1,h2,…,hn .Hãy xác định hình chữ nhật

có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.\*/

//PHAN HOÀNG NAM - 20215434

#include<bits/stdc++.h>

using namespace std;

long long findLargestRectArea(int n, int heights[]){

stack<int>S;//stack để lưu chỉ số của hcn

int maxArea = 0;//diện tích lớn nhất

int top;//biến lưu đỉnh stack

int area\_top\_as\_smallest;//diện tích lớn nhất của hcn với chiều cao là heights[top]

int i=0;//index hcn

while(i<n){//lặp

if(S.empty()||heights[i]>=heights[S.top()]){

//nếu stack rỗng hoặc gặp hcn thấp hơn hcn có chỉ số ở đỉnh stack

S.push(i++);//đẩy vào stack

}//PHAN HOÀNG NAM - 20215434

else{

top = S.top();//top = chỉ số hcn đỉnh stack

S.pop();//xóa phần tử đỉnh stack

int width = (S.empty()?i:i-S.top()-1);

//chiều rộng của hcn là i nếu stack rỗng

//là i-S.top()-1 nếu stack không rỗng

//i là chỉ số hiện tại (bên phải nhất)

//S.top() là chỉ số bên trái nhất

area\_top\_as\_smallest = heights[top]\*width;//tính diện tích hcn

maxArea = max(maxArea, area\_top\_as\_smallest);//cập nhật diện tích max

}

}//PHAN HOÀNG NAM - 20215434

while(!S.empty()){//nếu stack chưa rỗng thì lặp lại các bước ở else

top = S.top();

S.pop();

int width = (S.empty()?i:i-S.top()-1);

area\_top\_as\_smallest = heights[top]\*width;

maxArea = max(maxArea, area\_top\_as\_smallest);

}

return maxArea;

}//PHAN HOÀNG NAM - 20215434

int main(){

printf("PHAN HOANG NAM - 20215434\n");

freopen("input.txt", "r", stdin);

int n;

cin>>n;//nhập số hcn

int heights[n];//mảng chiều cao hcn

for(int i=0; i<n; i++){

cin>>heights[i];//nhập chiều cao hcn

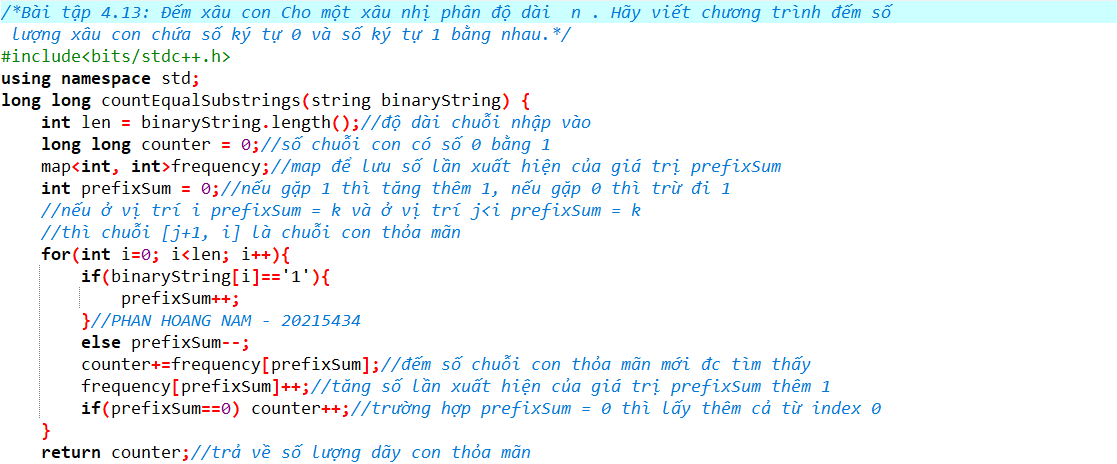
}

long long result = findLargestRectArea(n, heights);//result lưu kết quả trả về

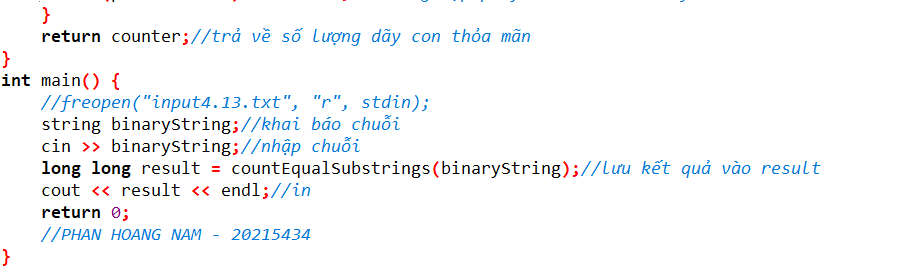
cout<<result;//in ra result

}

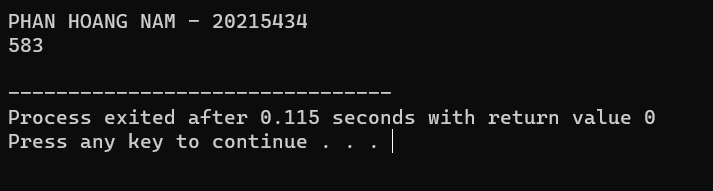
## Bài tập 4.13: Đếm xâu con Cho một xâu nhị phân độ dài n . Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.



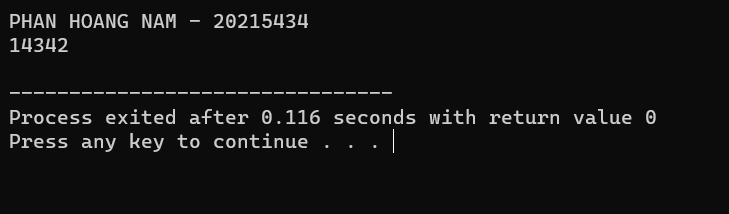
Hình 78. Code bài 4.13 ảnh 1



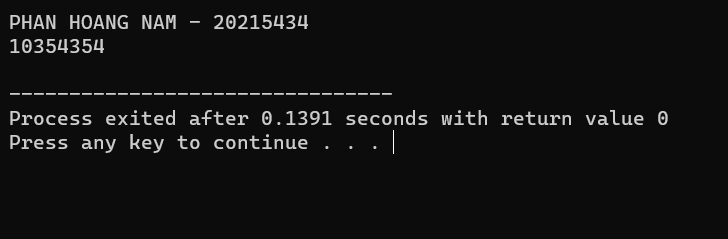
Hình 79. Code bài 4.13 ảnh 2



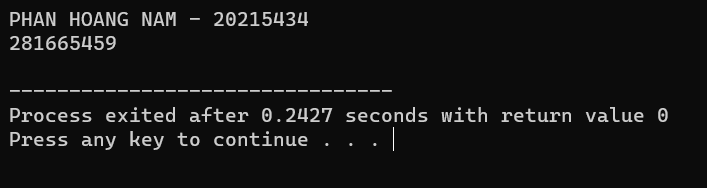
Hình 80. Test case 1 bài 4.13



Hình 81. Test case 2 bài 4.13



Hình 82. Test case 3 bài 4.13



Hình 83. Test case 4 bài 4.13

/\*Bài tập 4.13: Đếm xâu con Cho một xâu nhị phân độ dài n . Hãy viết chương trình đếm số

lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.\*/

#include<bits/stdc++.h>

using namespace std;

long long countEqualSubstrings(string binaryString) {

long long len = binaryString.length();//độ dài chuỗi nhập vào

long long counter = 0;//số chuỗi con có số 0 bằng 1

map<long long, long long>frequency;//map để lưu số lần xuất hiện của giá trị prefixSum

long long prefixSum = 0;//nếu gặp 1 thì tăng thêm 1, nếu gặp 0 thì trừ đi 1

//nếu ở vị trí i prefixSum = k và ở vị trí j<i prefixSum = k

//thì chuỗi [j+1, i] là chuỗi con thỏa mãn

for(int i=0; i<len; i++){

if(binaryString[i]=='1'){

prefixSum++;

}//PHAN HOANG NAM - 20215434

else prefixSum--;

counter+=frequency[prefixSum];//đếm số chuỗi con thỏa mãn mới đc tìm thấy

frequency[prefixSum]++;//tăng số lần xuất hiện của giá trị prefixSum thêm 1

if(prefixSum==0) counter++;//trường hợp prefixSum = 0 thì lấy thêm cả từ index 0

}

return counter;//trả về số lượng dãy con thỏa mãn

}

int main() {

printf("PHAN HOANG NAM - 20215434\n");

freopen("input4.13.txt", "r", stdin);

string binaryString;//khai báo chuỗi

cin >> binaryString;//nhập chuỗi

long long result = countEqualSubstrings(binaryString);//lưu kết quả vào result

cout << result << endl;//in

return 0;

//PHAN HOANG NAM - 20215434

}