**BÁO CÁO BUỔI THỰC HÀNH SỐ 4**

Bộ Môn Kỹ Thuật Lập Trình

A blue logo with black background

Description automatically generated

A red and white logo

Description automatically generated

Sinh viên: MSSV-Họ và tên

Giáo viên hướng dẫn: ThS. Lê Thị Hoa

HTGD: Đoàn Sỹ Nguyên

*Contents*

[*Bài thực hành số 4 – Tuần 14 3*](#_Toc153398883)

[***Bài 4.1:****Đảo ngược một danh sách liên kết đơn 3*](#_Toc153398884)

[***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*](#_Toc153398885)

[***Bài 4.3****: Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm tính tích có hướng của 2 vector. 9*](#_Toc153398886)

[***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. 10*](#_Toc153398887)

[***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). 12*](#_Toc153398888)

[***Bài 4****. đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật 6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên 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). 16*](#_Toc153398889)

[***Bài 4.7****. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set 20*](#_Toc153398890)

[***Bài 4.8.****Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. 23*](#_Toc153398891)

[***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 28*](#_Toc153398892)

[***Bài10:*** *Search Engine 33*](#_Toc153398893)

[***Bài11.*** *Bảo vệ lâu đài 36*](#_Toc153398894)

[***Bài 12****. Lược đồ 38*](#_Toc153398895)

[***Bài 13:*** *Đếm xâu con 39*](#_Toc153398896)

[Hình 1. 1.Code bài 1. 3](#_Toc153396976)

[Hình 1. 2.Testcase bài 1 3](#_Toc153396977)

[Hình 2. 1.Code bài 2. 7](#_Toc153396985)

[Hình 2. 2. Testcase bài 2. 7](#_Toc153396986)

[Hình 3. 1 Code bài 3 9](#_Toc153396991)

[Hình 3. 2 Testcase bài 3 9](#_Toc153396992)

[Hình 4. 1 Code bài 4 11](#_Toc153396998)

[Hình 4. 2 Testcase bài 4 11](#_Toc153396999)

[Hình 5. 1. Cod bài 5 12](#_Toc153397003)

[Hình 5. 2 Testcase bài 5 13](#_Toc153397004)

[Hình 6. 1 Code bài 6 16](#_Toc153397009)

[Hình 6. 2 Testacse bài 6 17](#_Toc153397010)

[Hình 7. 1 Code bài 7 20](#_Toc153397014)

[Hình 7. 2 Testcase bài 7 21](#_Toc153397015)

[Hình 8. 1 Code bài 8 24](#_Toc153397018)

[Hình 8. 2 TestCase bài 8 24](#_Toc153397019)

[Hình 9. 1 Code bài 9 29](#_Toc153397022)

[Hình 9. 2 Testcase bài 9 30](#_Toc153397023)

[Hình 10. 1 Testcase bài 10 36](#_Toc153398937)

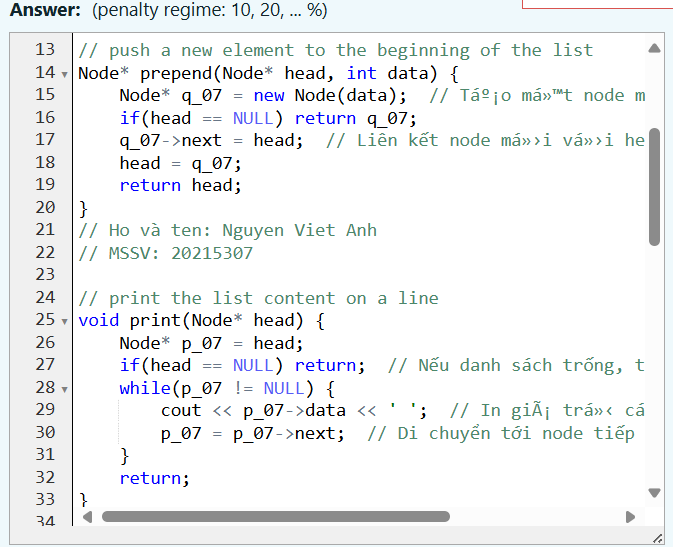
[Hình 11. 1 Test case bài 11 38](#_Toc153398943)

[Hình 12. 1.Testcase bài 12 39](#_Toc153398948)

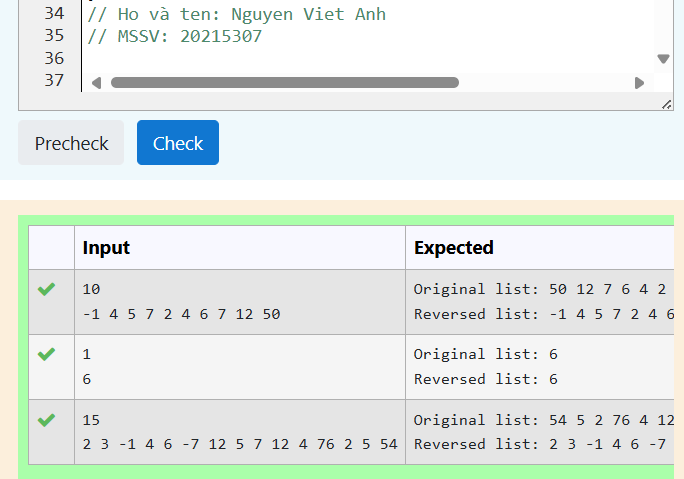
[Hình 13. 1 Test case bài 13 40](#_Toc153398951)

# Bài thực hành số 4 – Tuần 14

# **Bài 4.1:**Đảo ngược một danh sách liên kết đơn

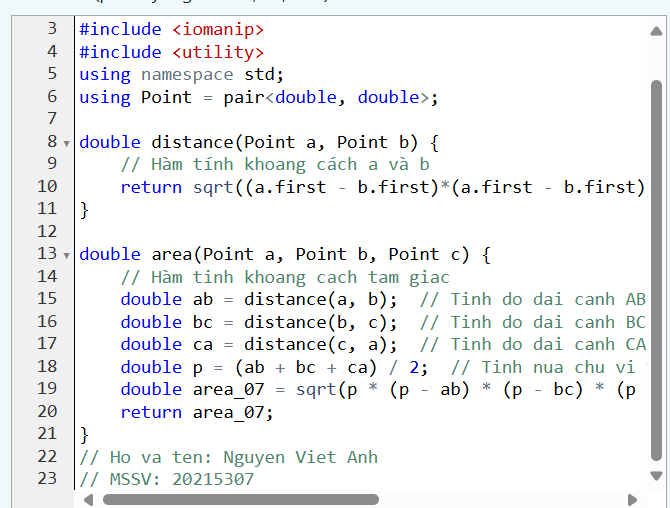


Hình 1. 1.Code bài 1.



Hình 1. 2.Testcase bài 1

# **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.

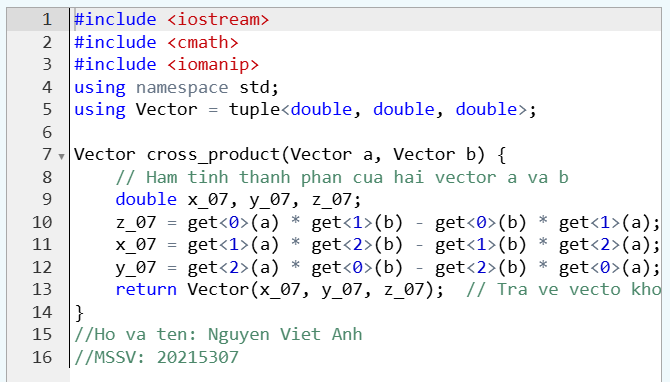


Hình 2. 1.Code bài 2.

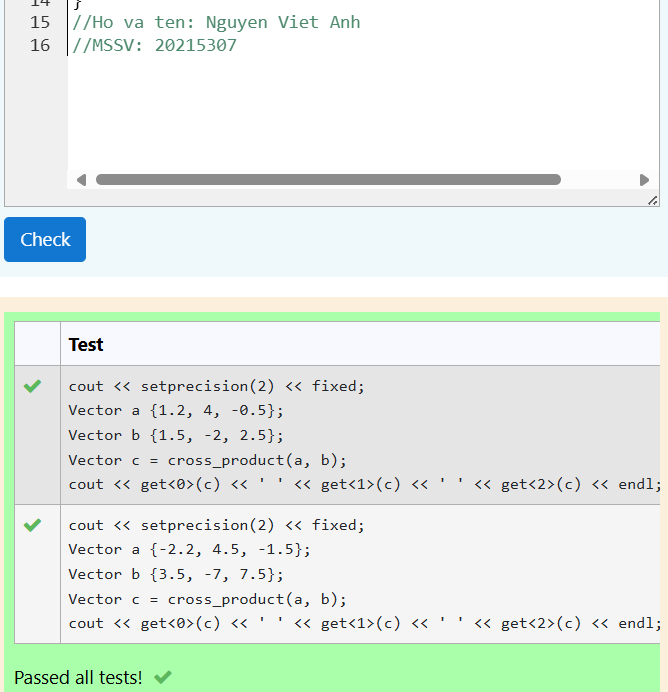


Hình 2. 2. Testcase bài 2.

# **Bài 4.3**: Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm tính tích có hướng của 2 vector.

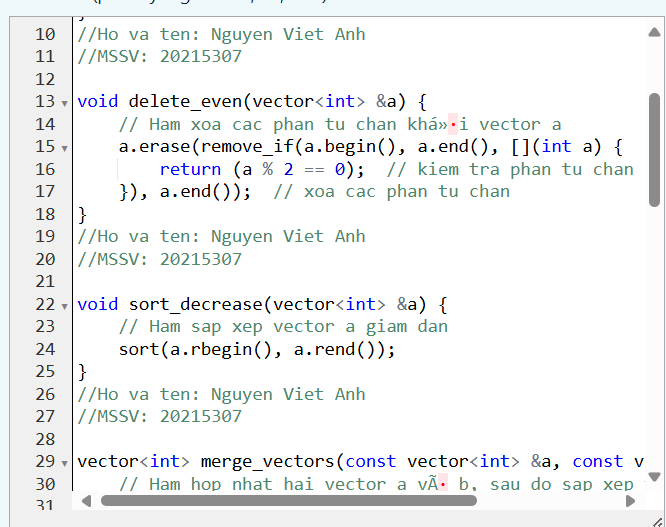


Hình 3. 1 Code bài 3

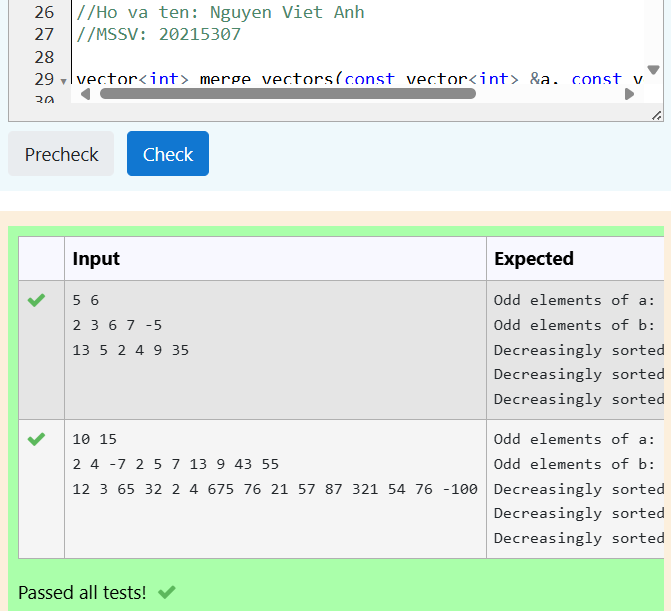


Hình 3. 2 Testcase bài 3

# **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.

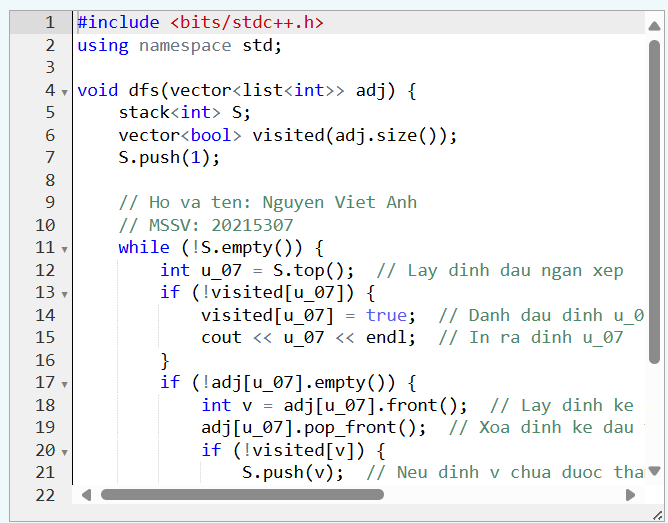


Hình 4. 1 Code bài 4

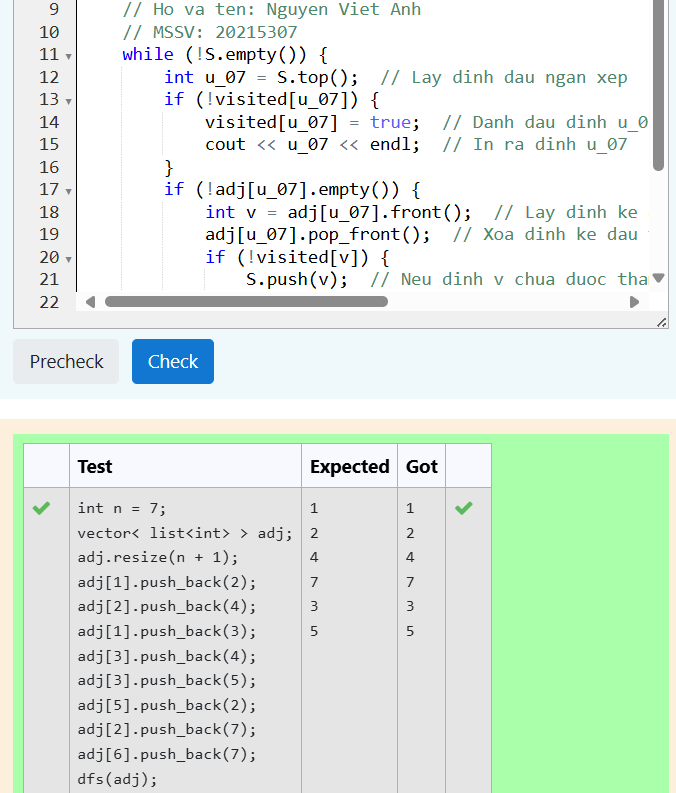


Hình 4. 2 Testcase bài 4

# **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).

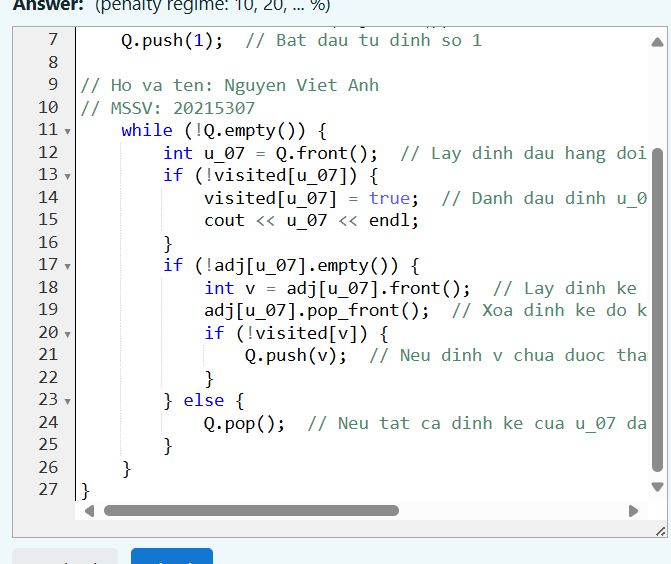


Hình 5. 1. Cod bài 5

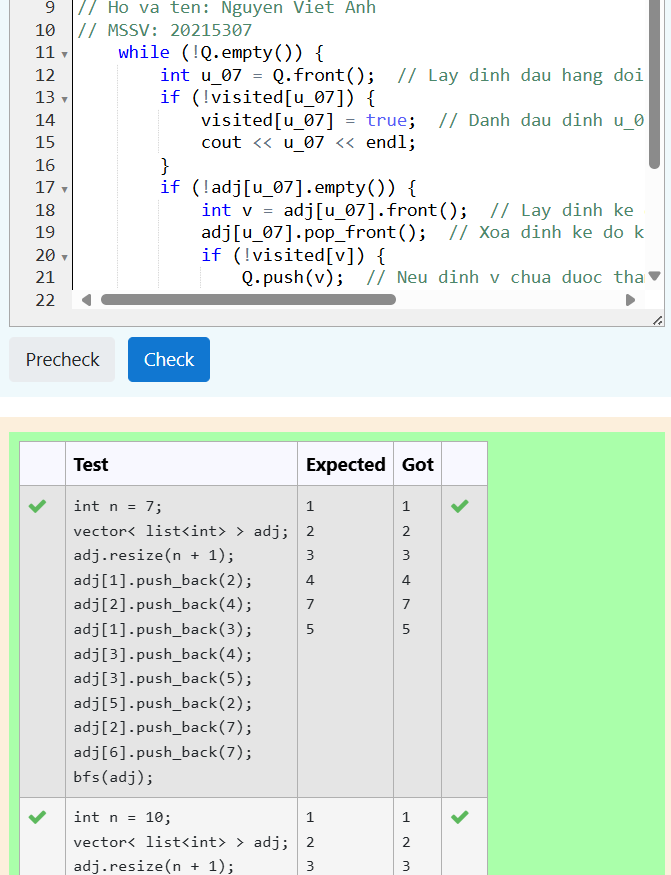


Hình 5. 2 Testcase bài 5

# **Bài 4.6**. đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật 6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên 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).

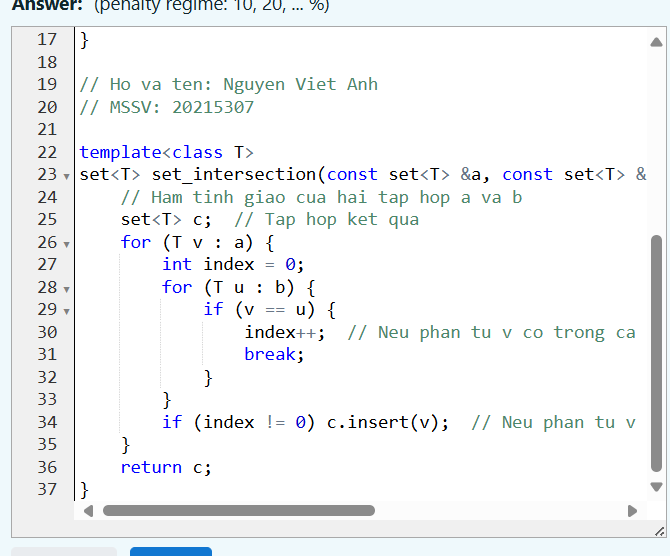


Hình 6. 1 Code bài 6

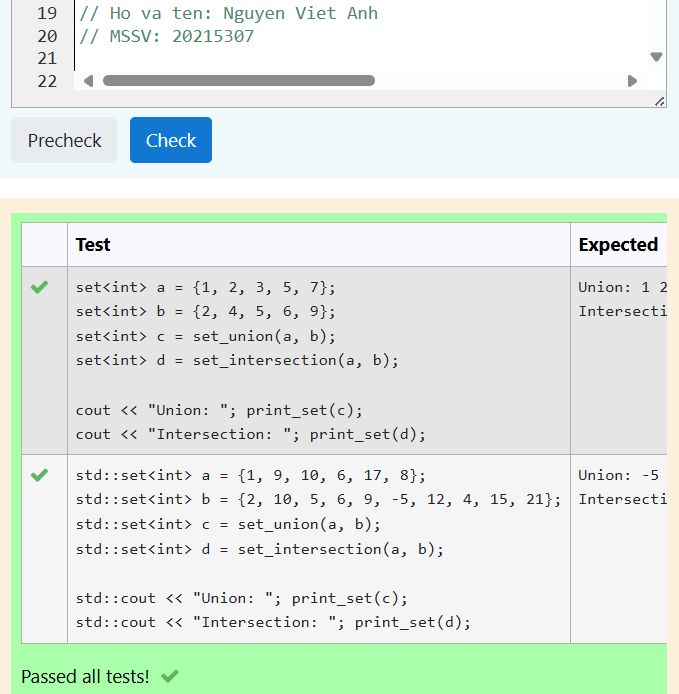


Hình 6. 2 Testacse bài 6

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

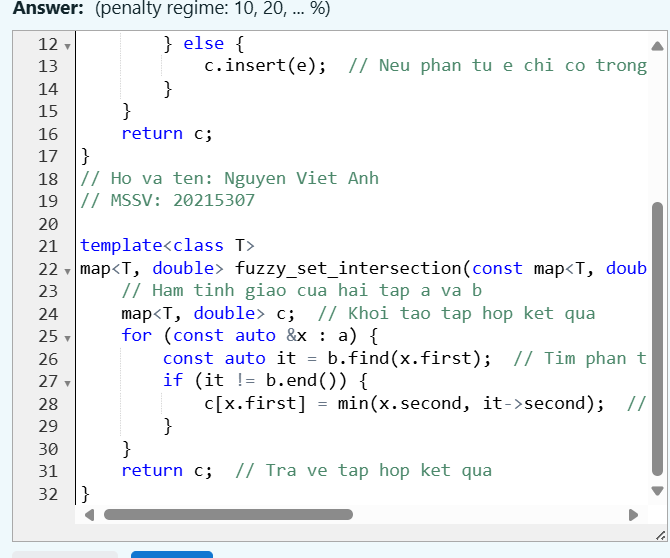


Hình 7. 1 Code bài 7

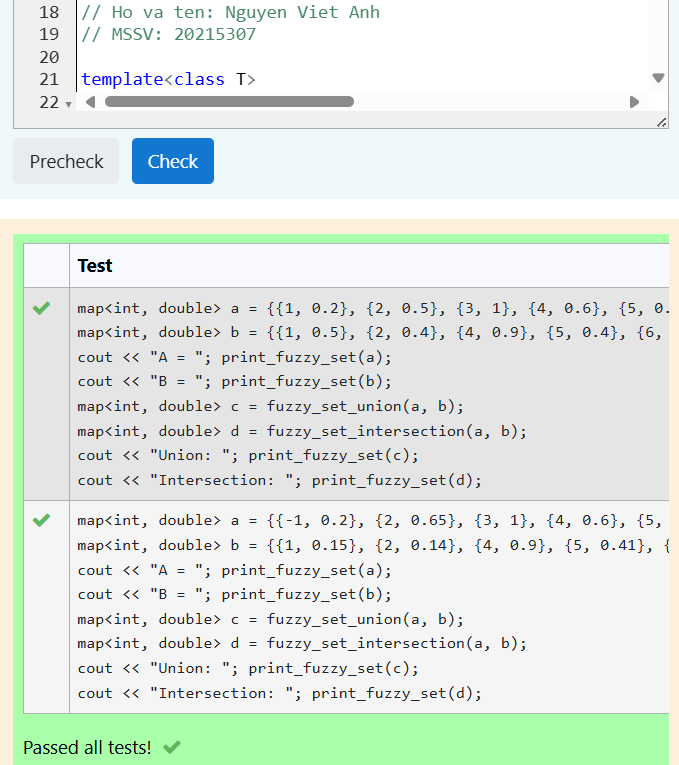


Hình 7. 2 Testcase bài 7

# **Bài 4.8.** Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.

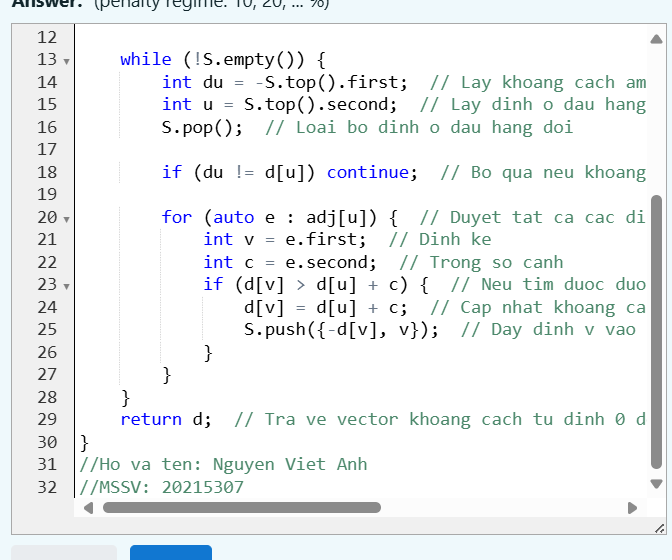


Hình 8. 1 Code bài 8

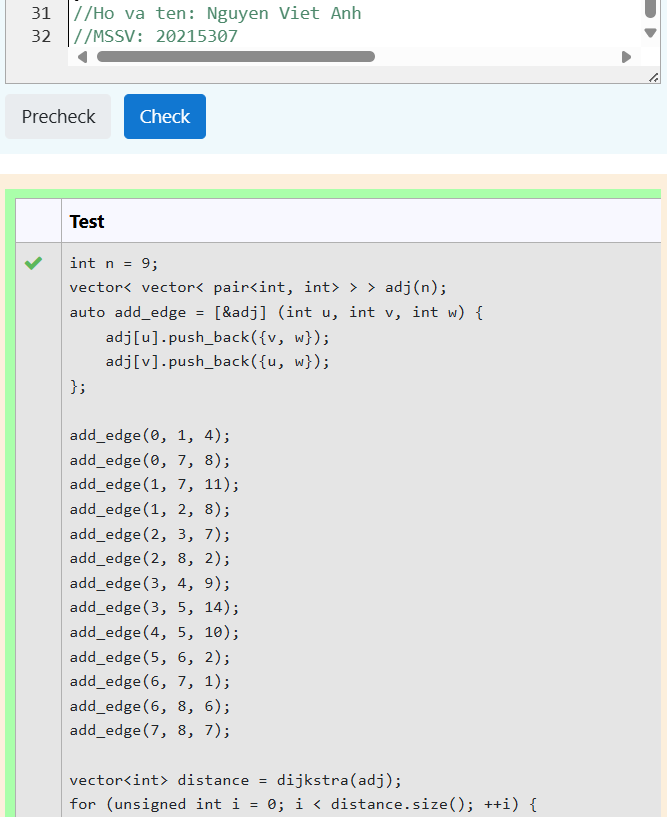


Hình 8. 2 TestCase bài 8

# **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



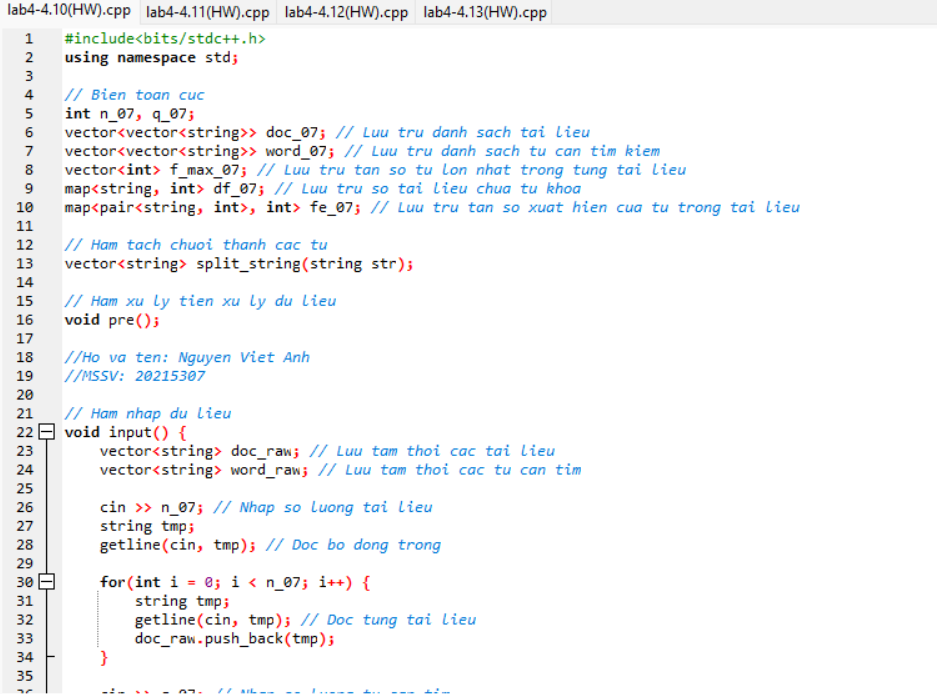
Hình 9. 1 Code bài 9

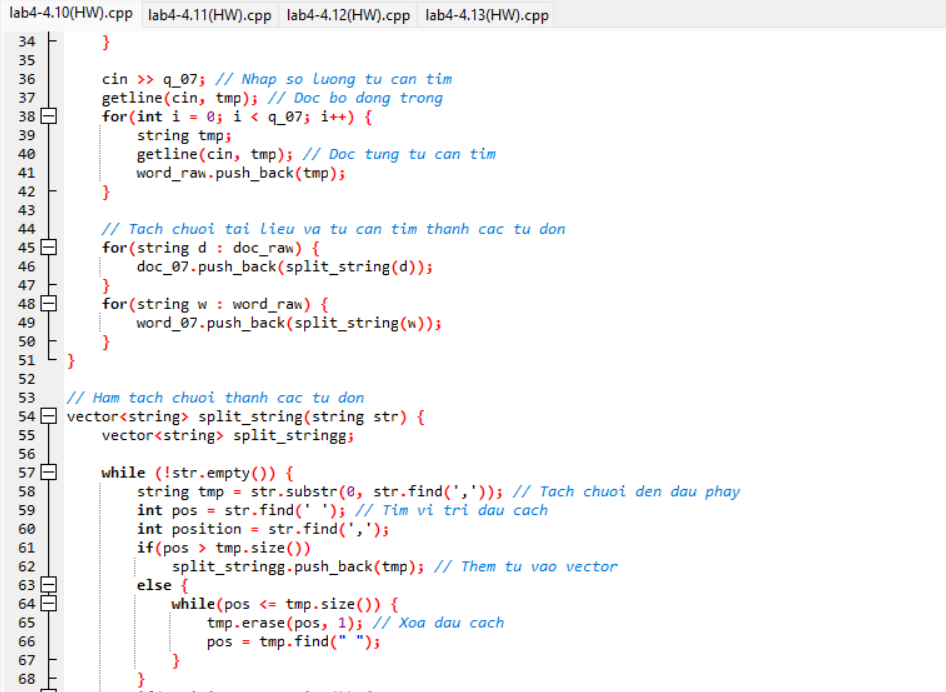


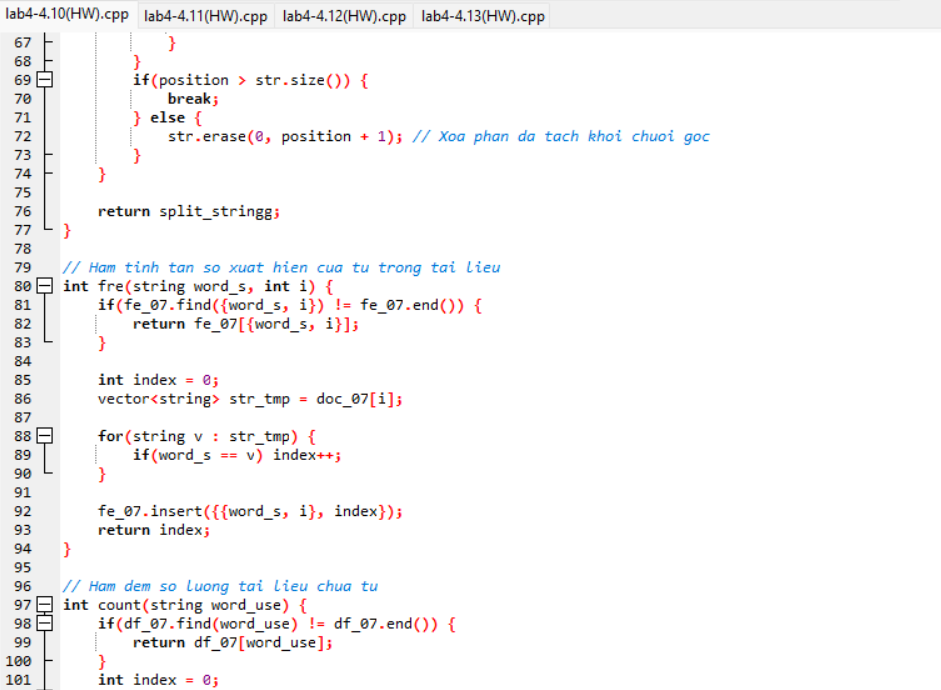
Hình 9. 2 Testcase bài 9

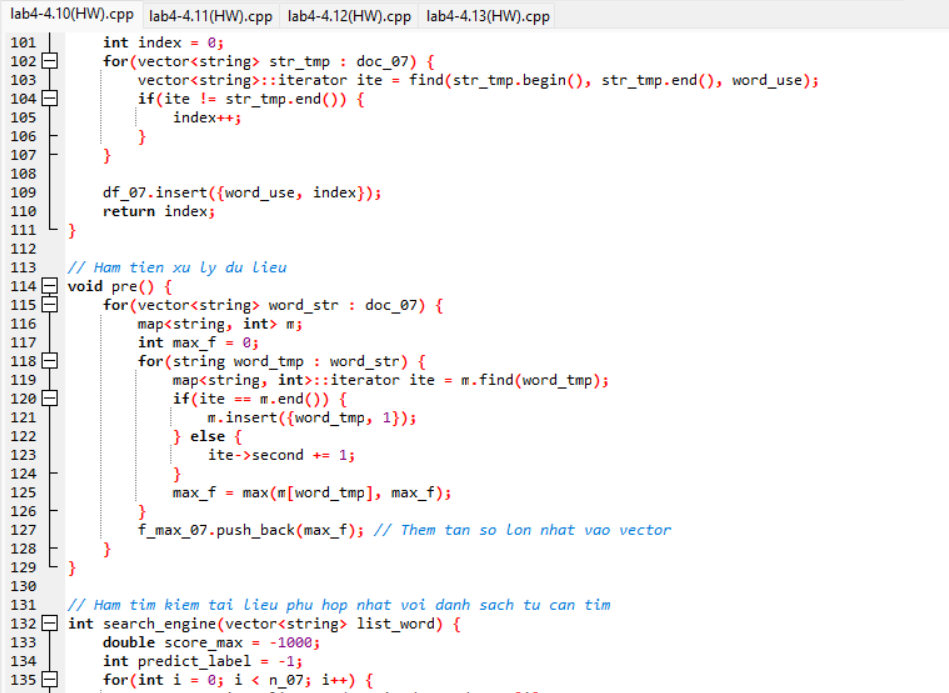
**BÀI TẬP VỀ NHÀ**

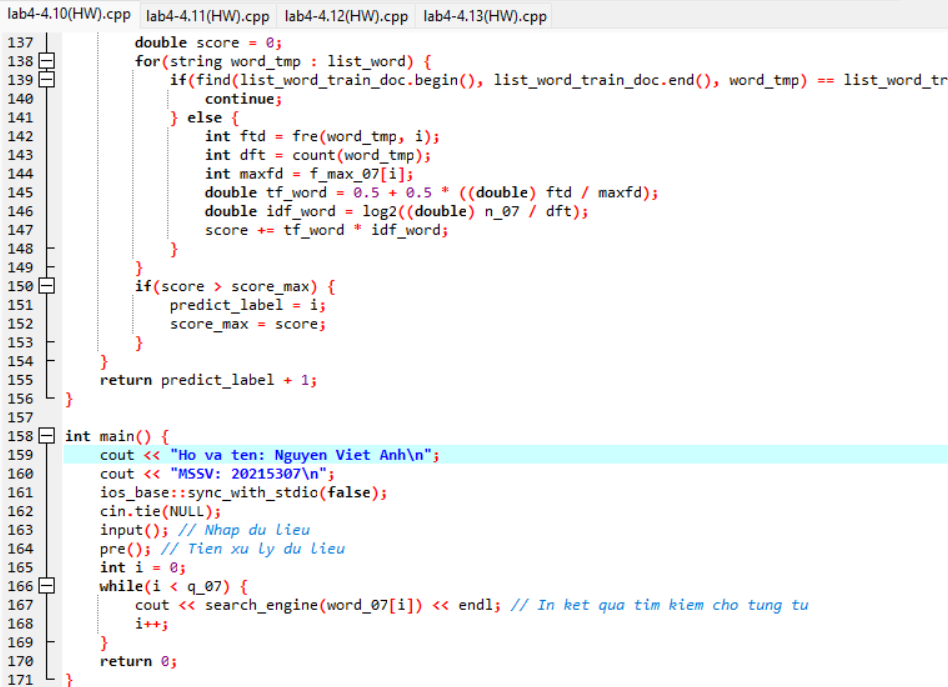
**Bài 10.**

****

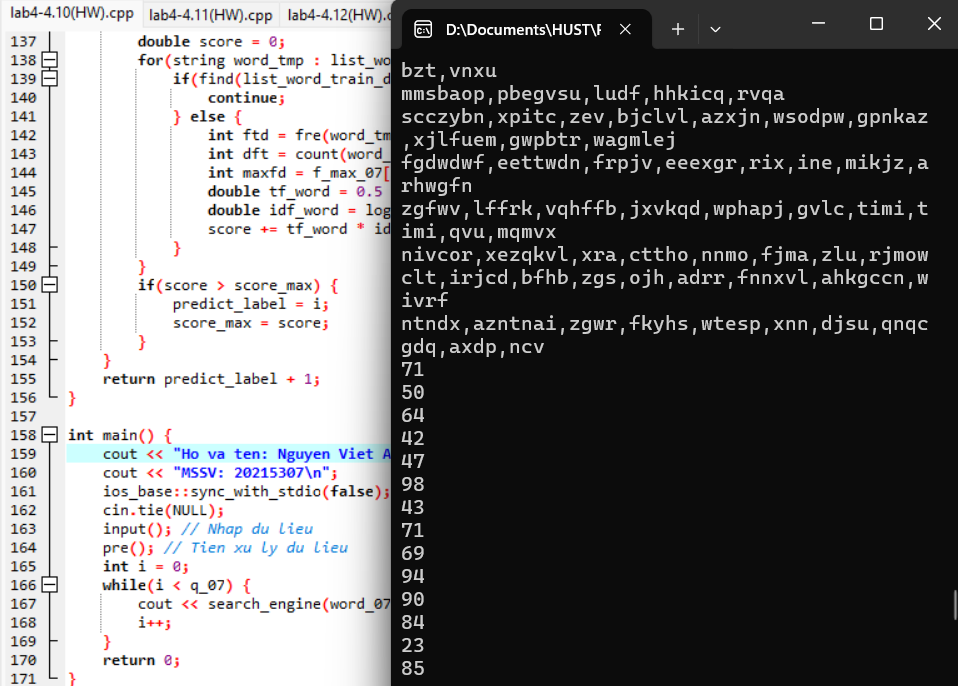
****

****

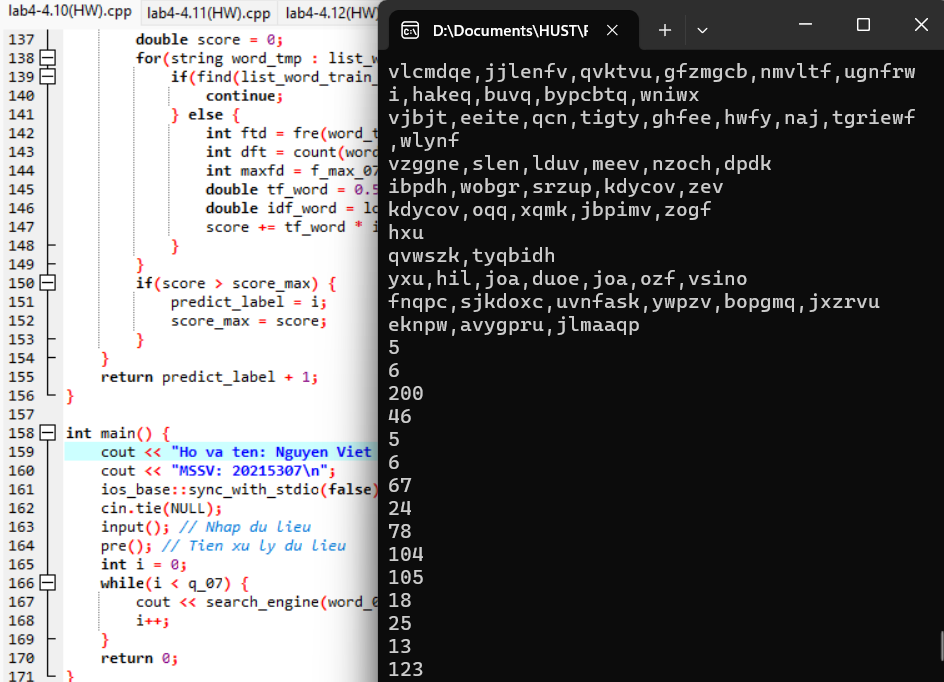
****

****

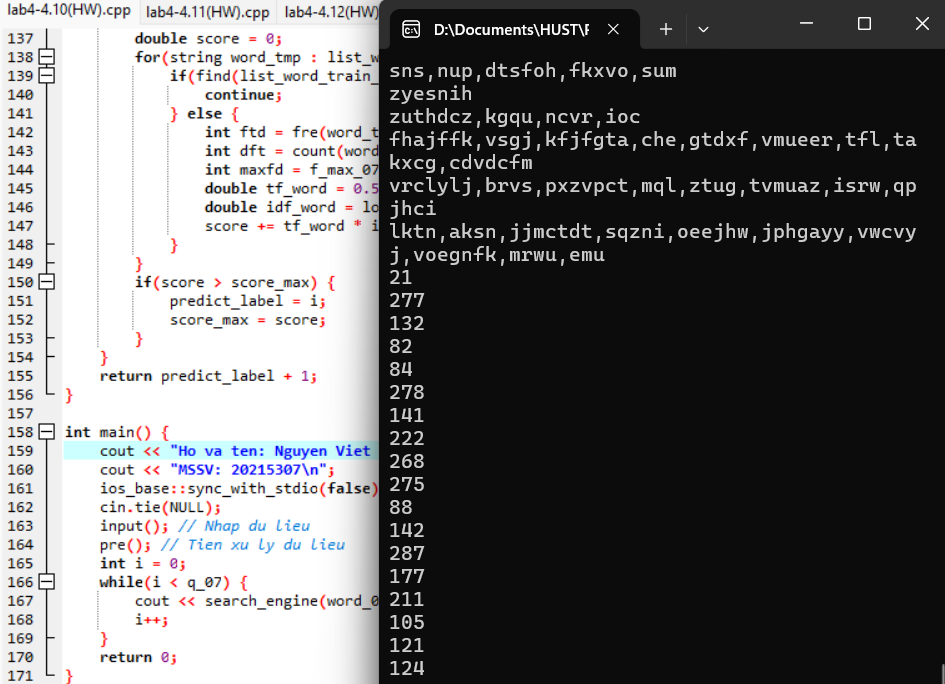
*Code bài 10*

**

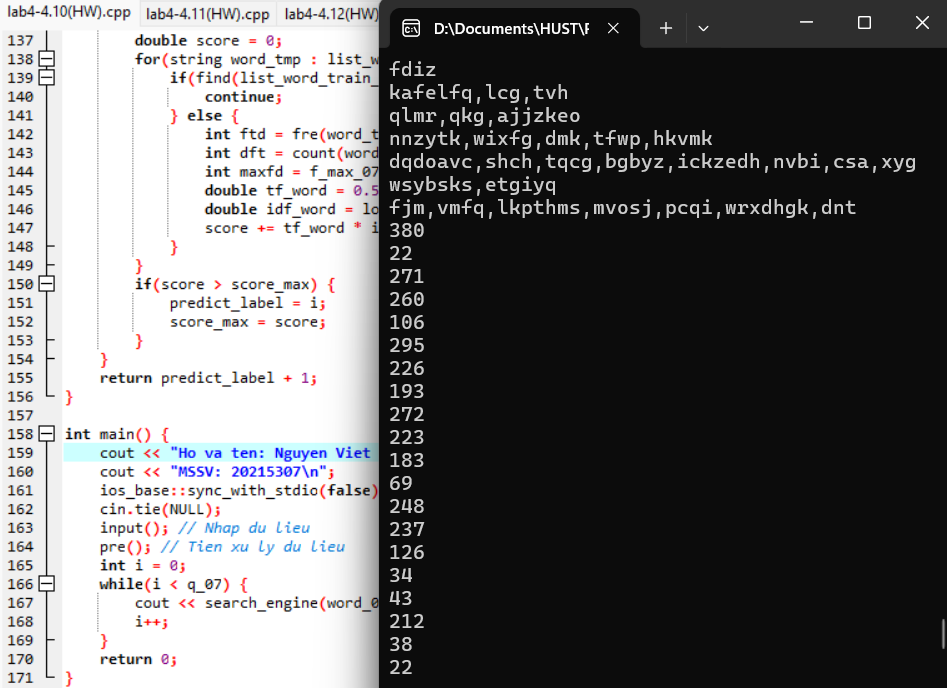
*Test case 1*

**

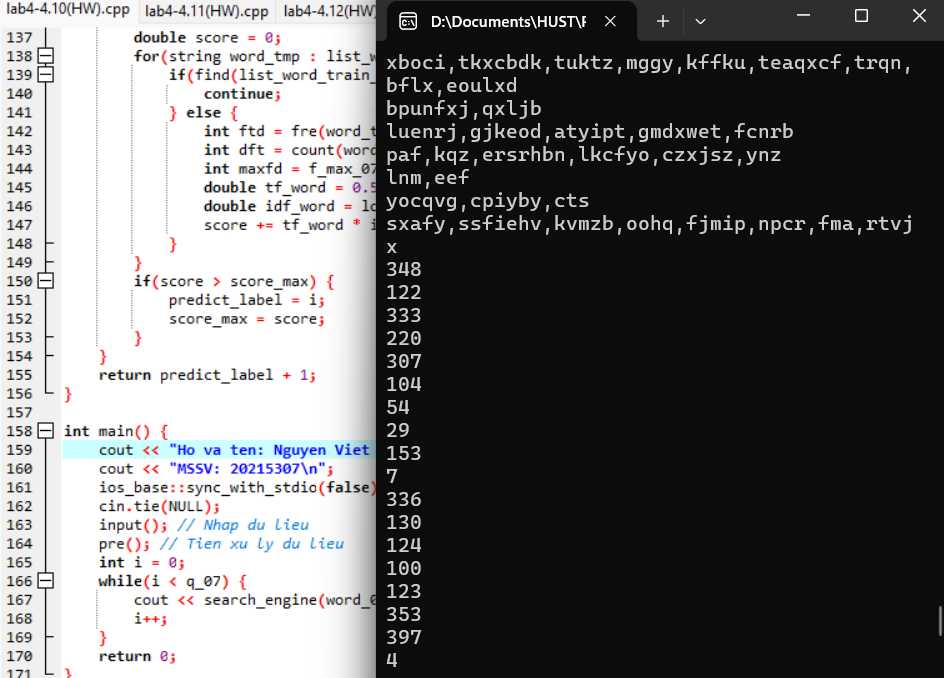
*Test case 2*

**

*Test case 3*

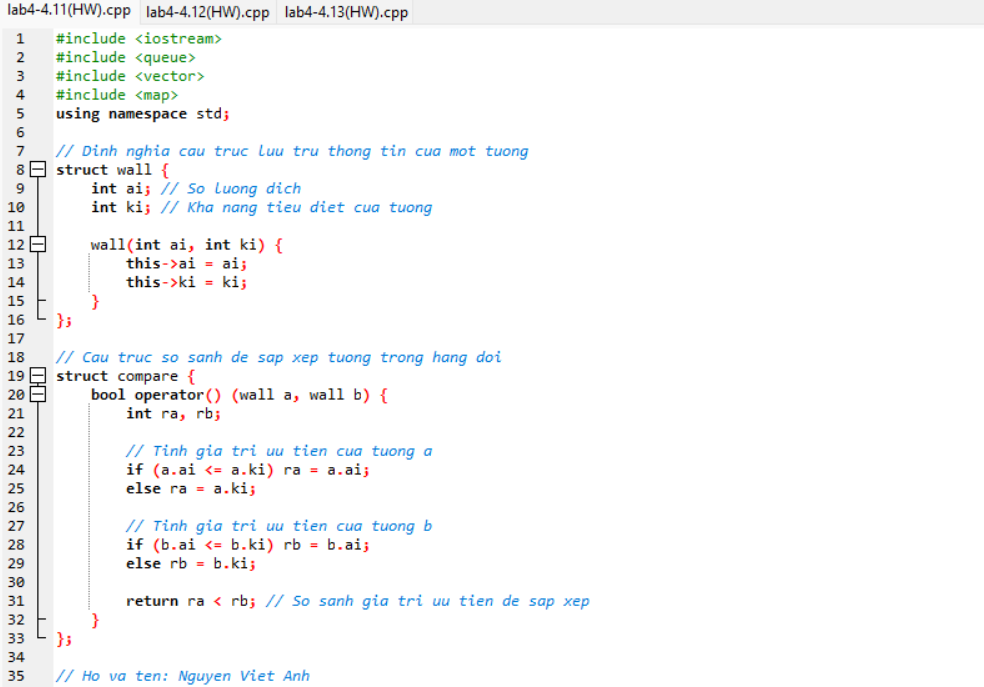
**

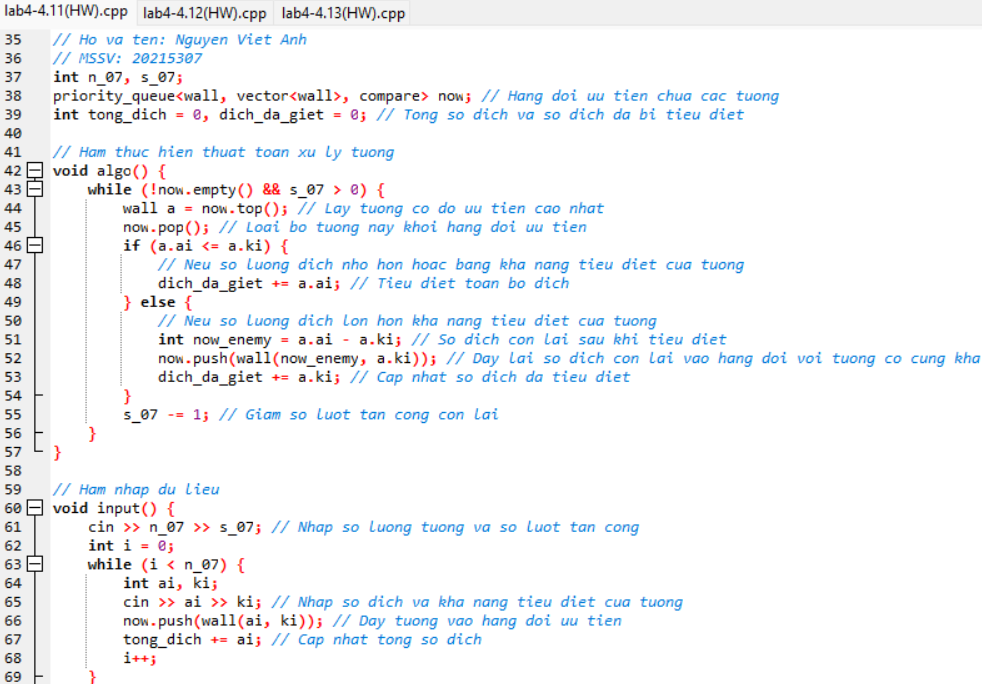
*Test case 4*

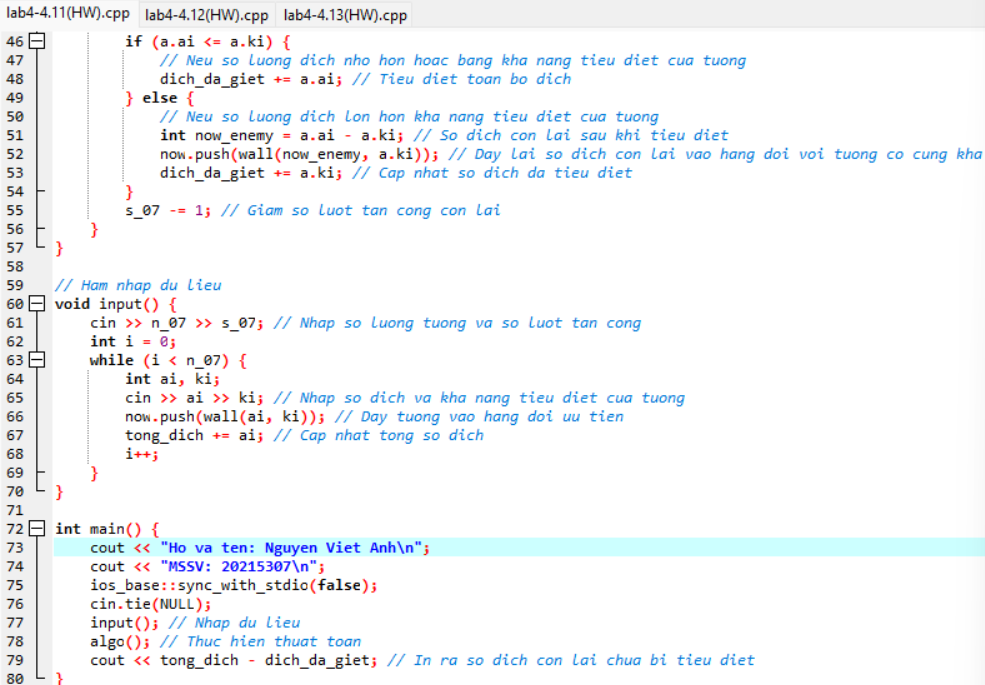
**

*Test case 5*

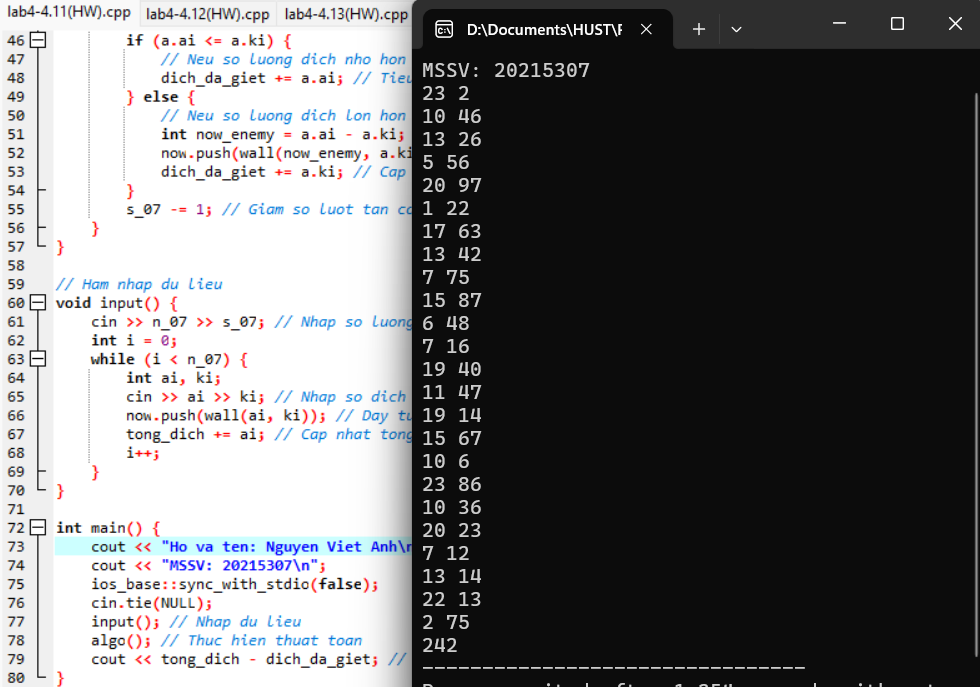
**Bài 11.**

**

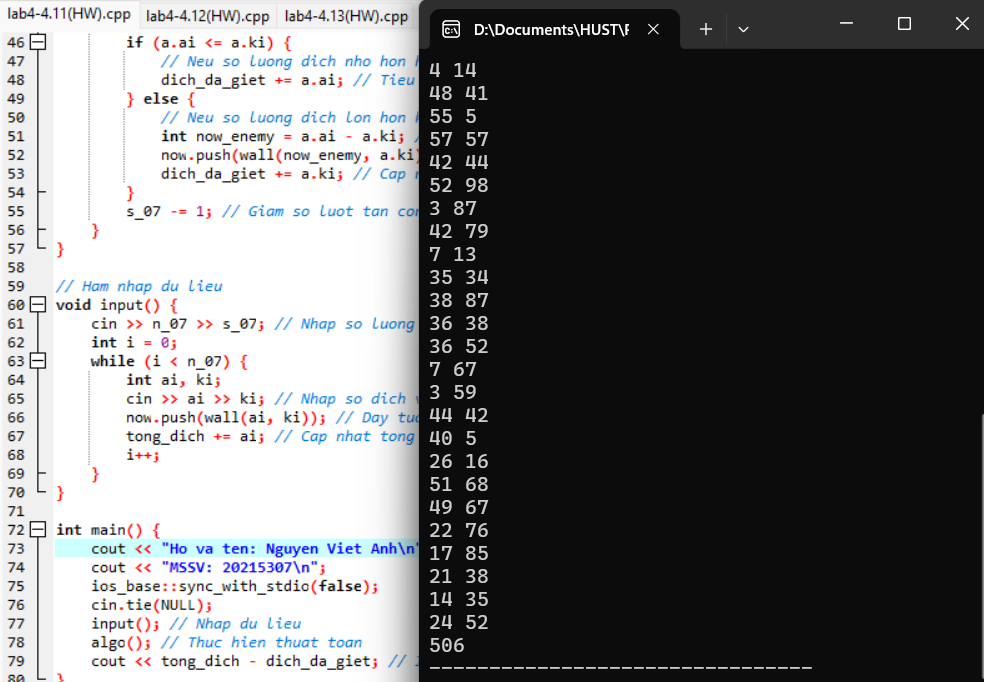
**

**

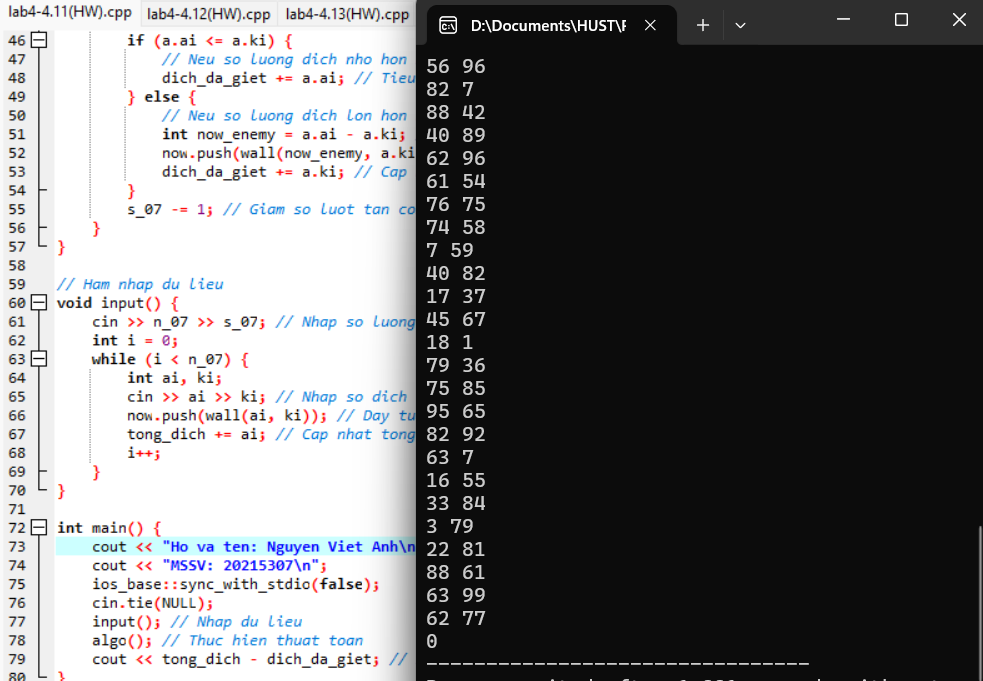
*Code bài 11*

**

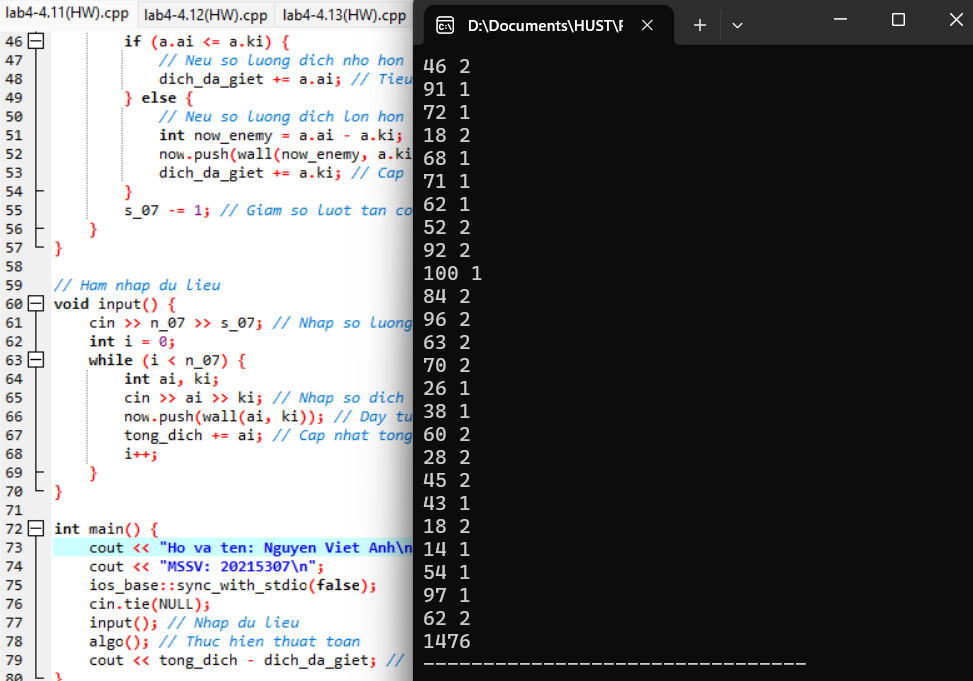
*Test case 1*

**

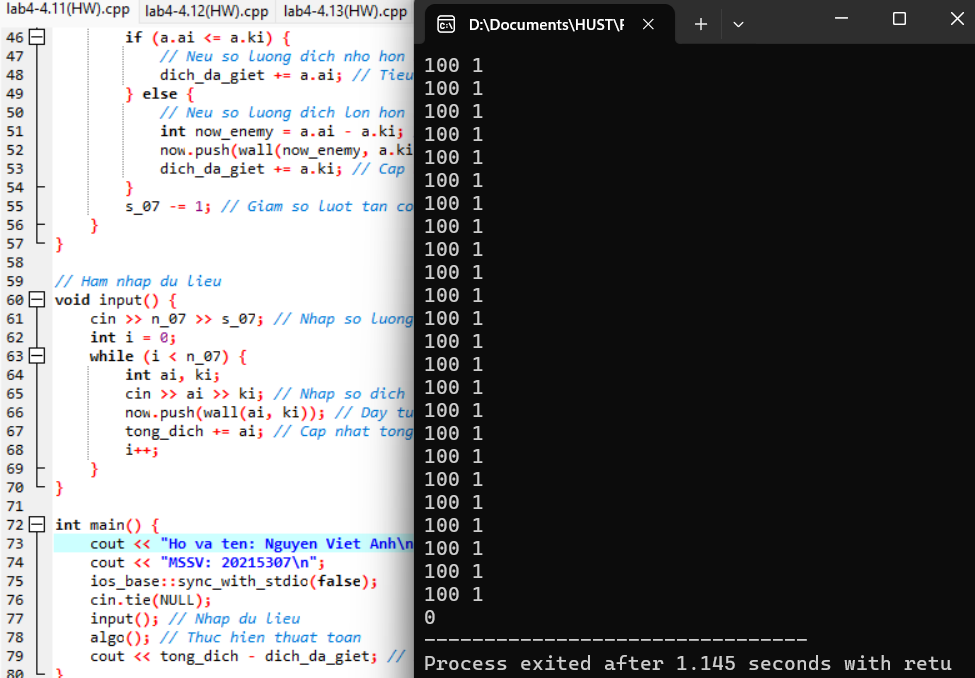
*Test case 2*

**

*Test case 3*

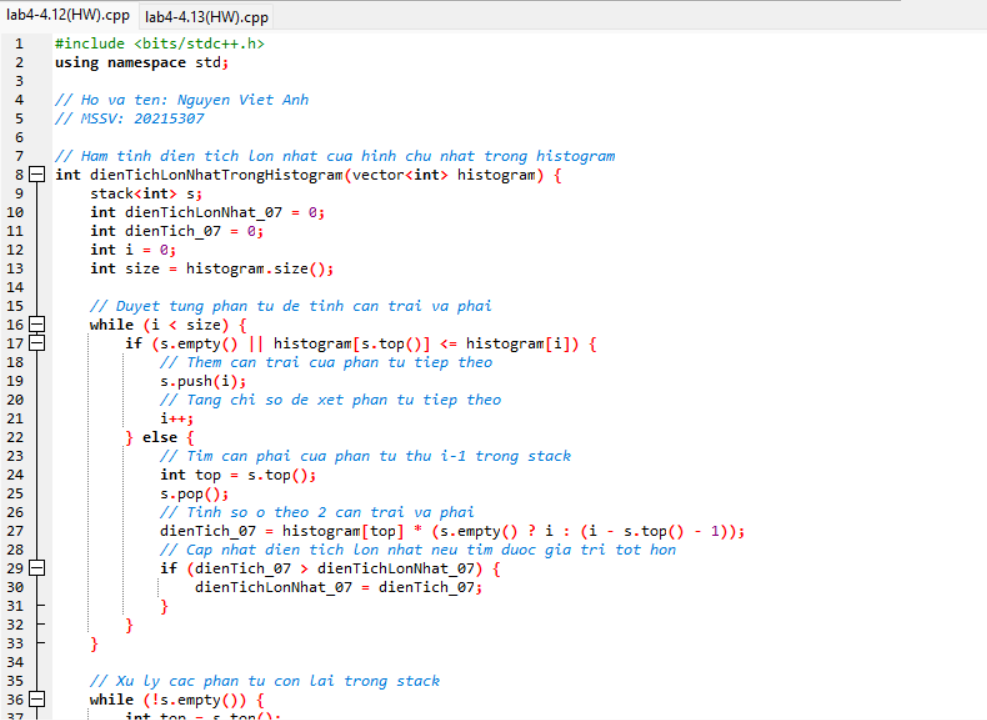
**

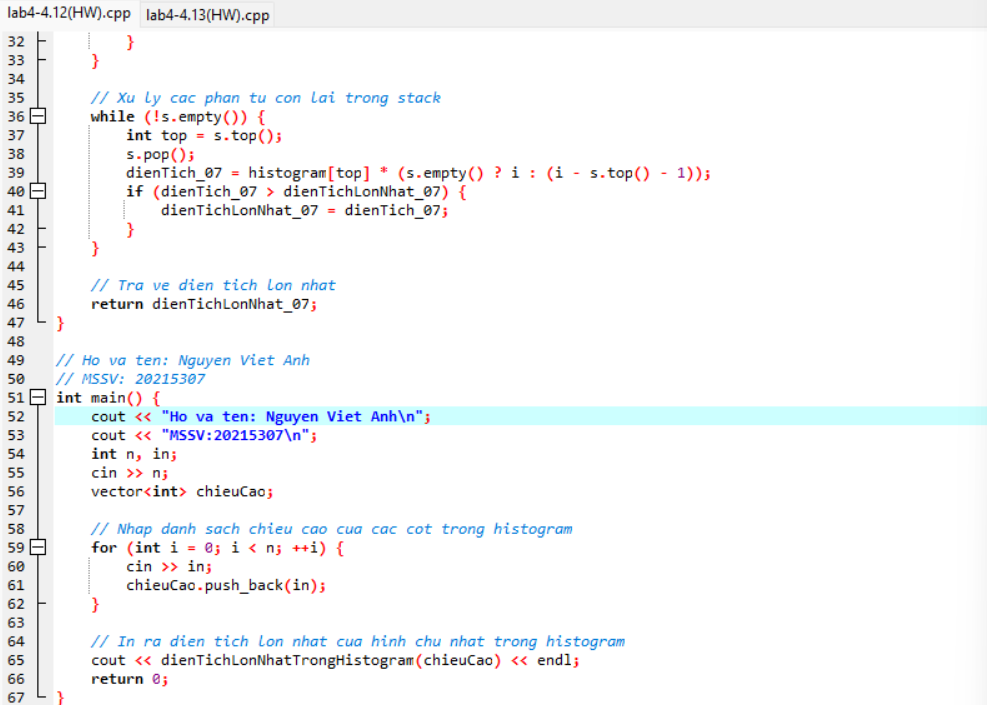
*Test case 4*

**

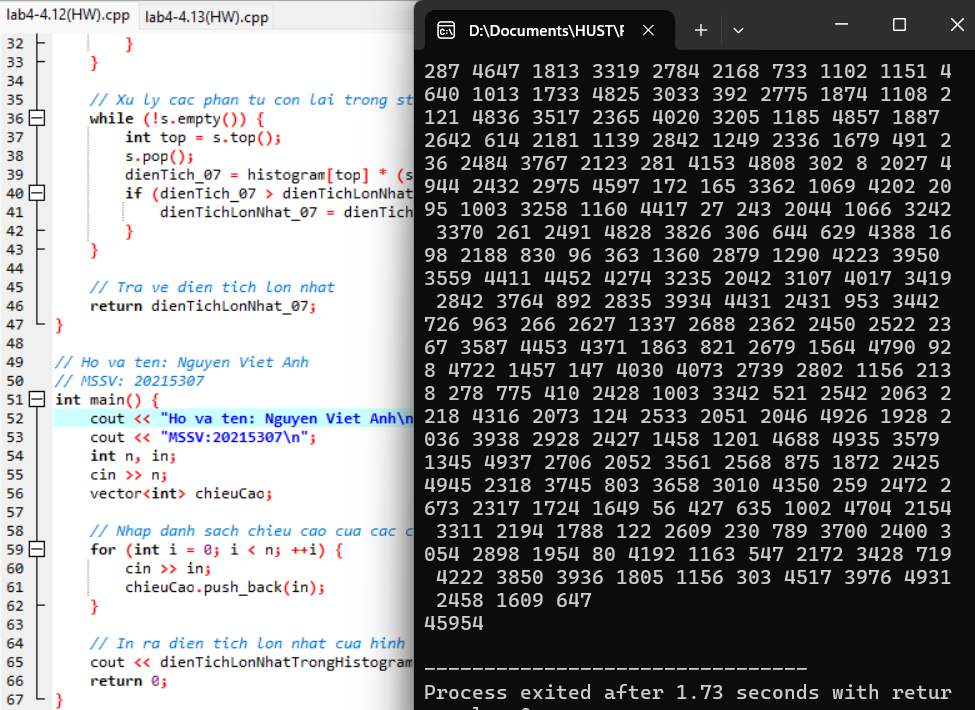
*Test case 5*

**Bài 12.**

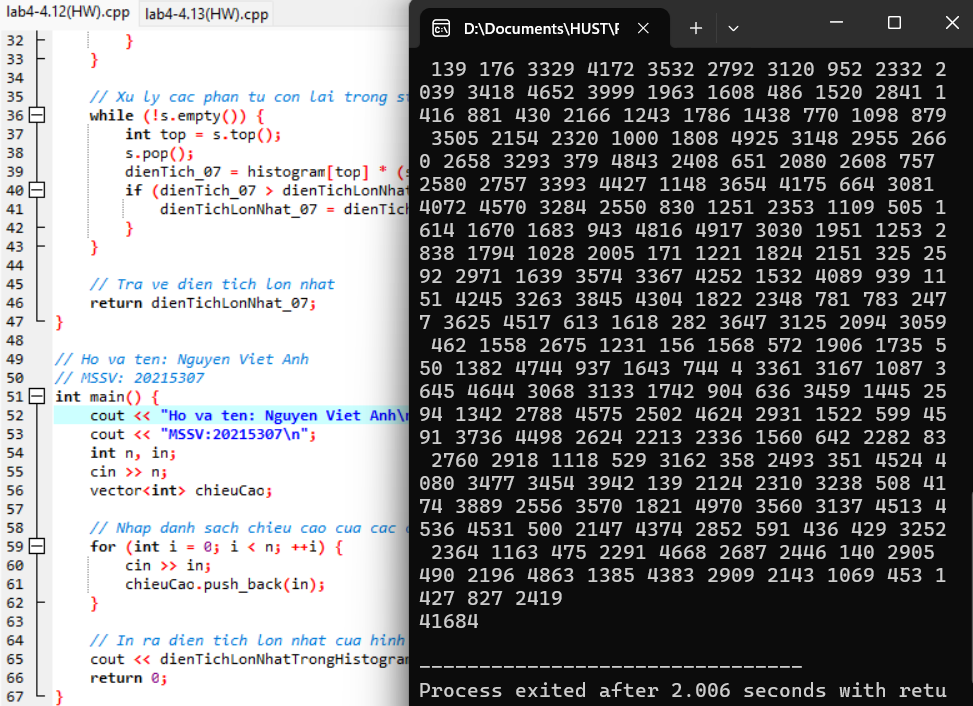
**

**

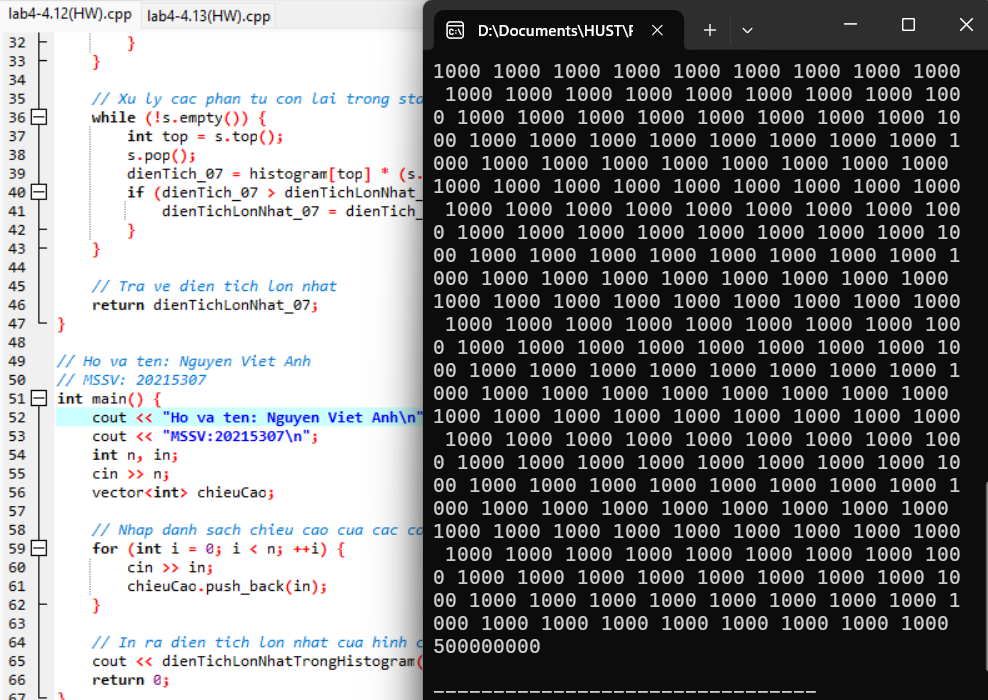
*Code bài 12*

**

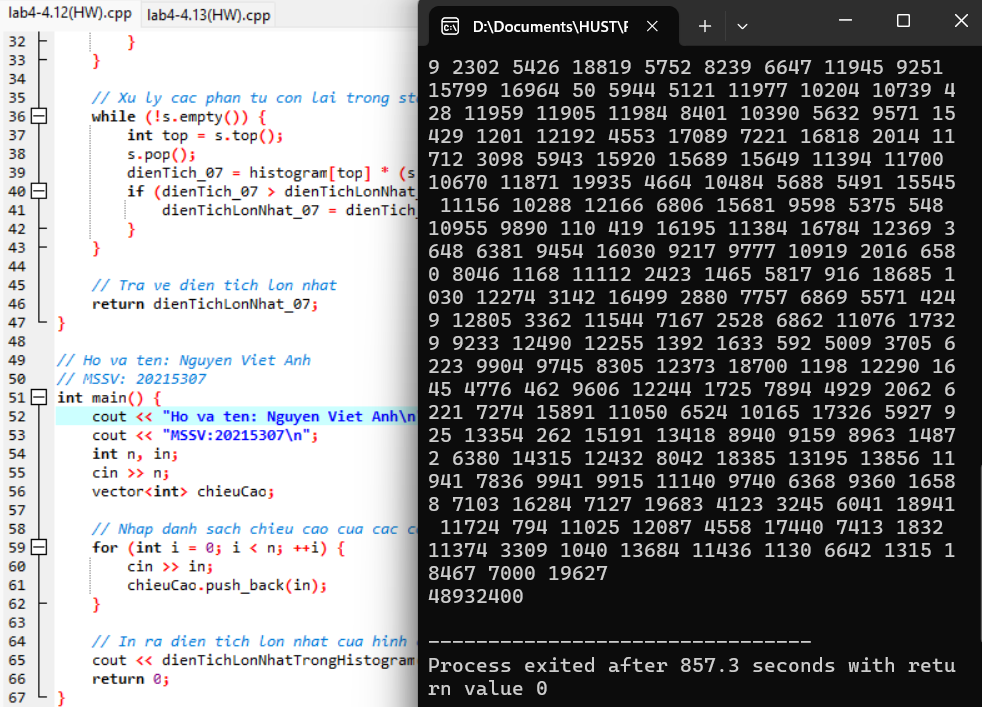
*Test case 1*

**

*Test case 2*

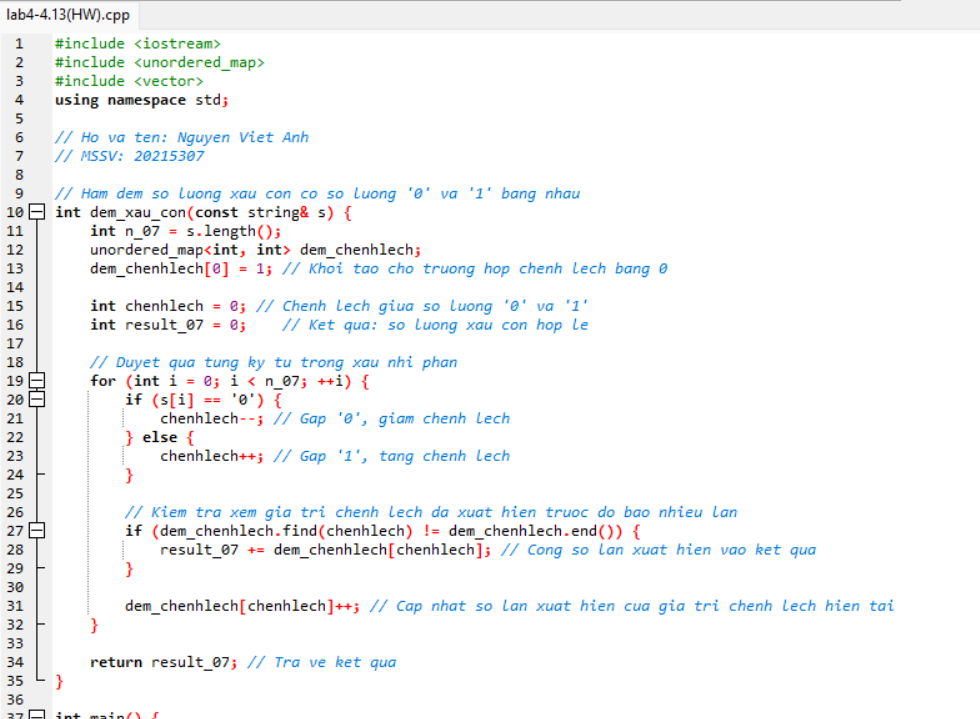
**

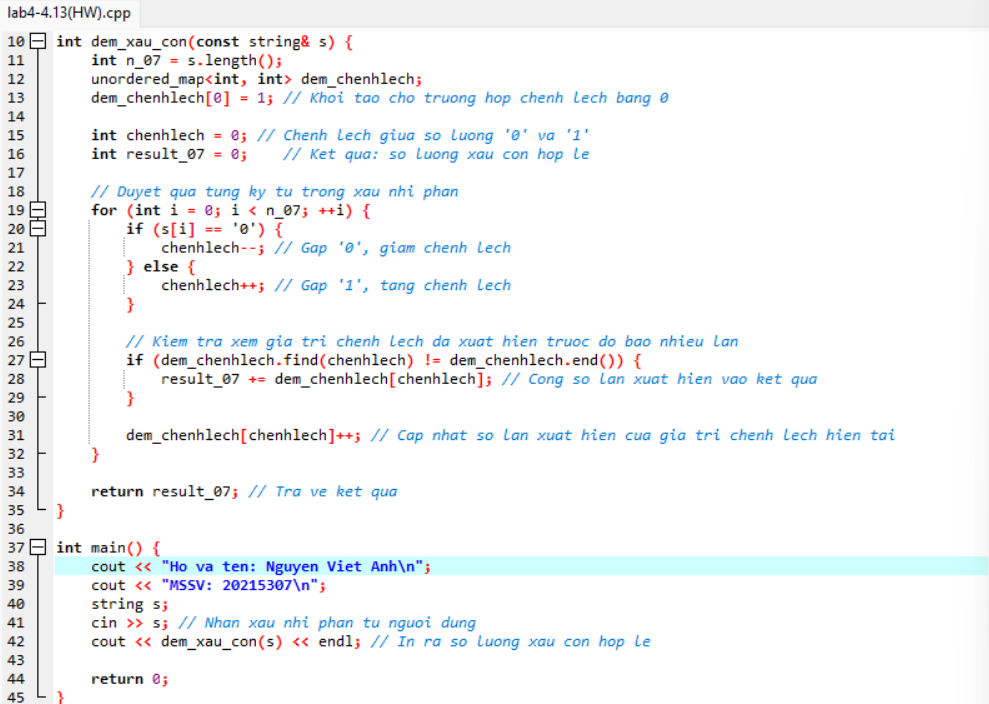
*Test case 3*

**

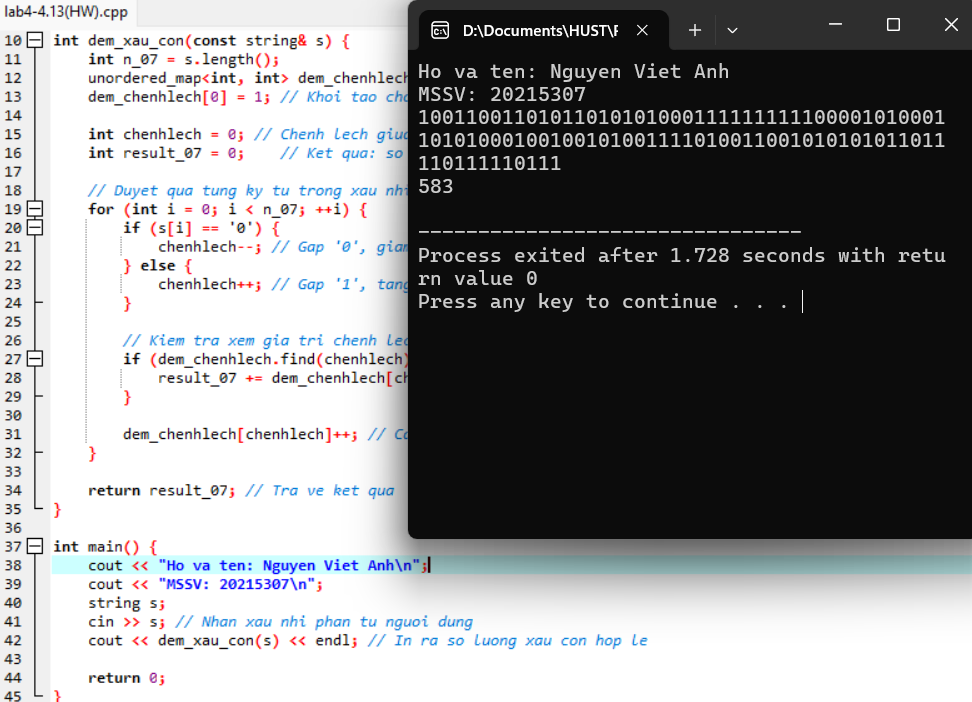
*Test case 4*

**Bài 13.**

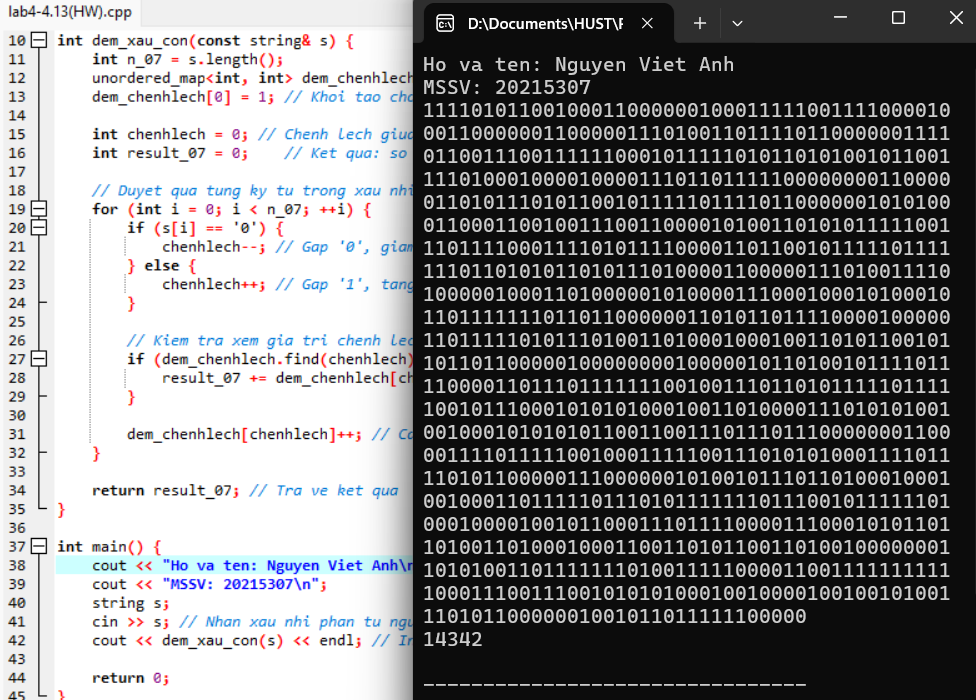
**

**

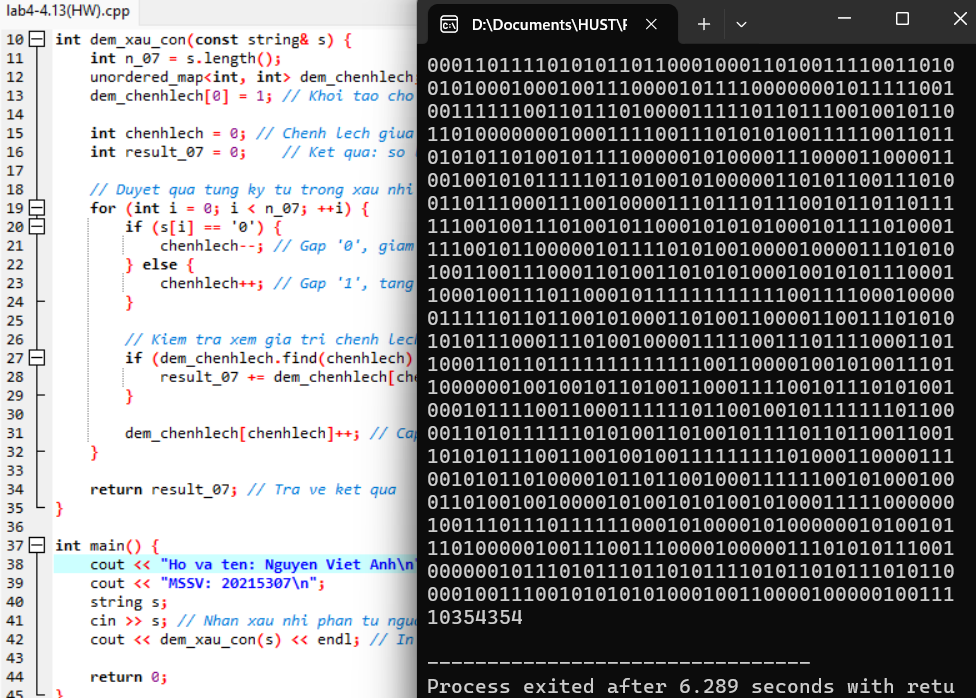
*Code bài 13*

**

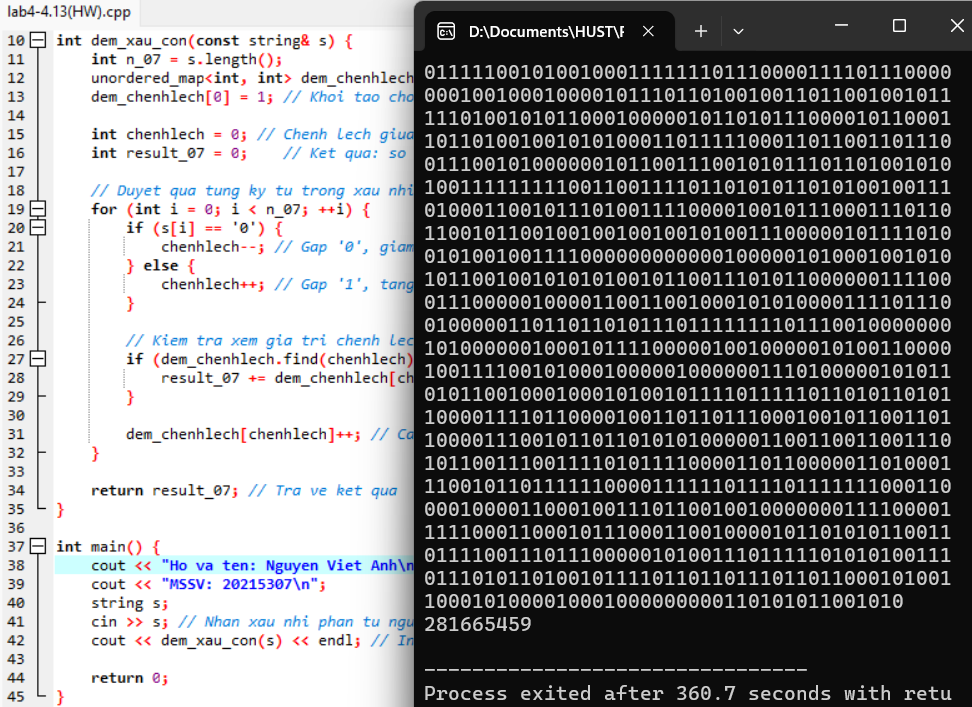
*Test case 1*

**

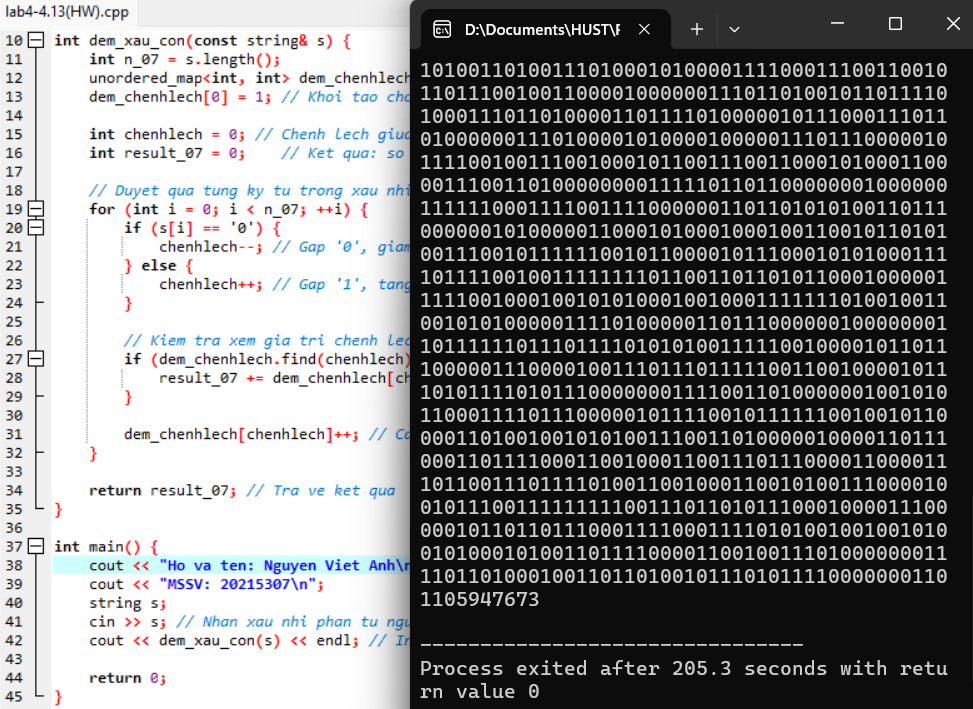
*Test case 2*

**

*Test case 3*

**

*Test case 4*

**

*Test case 5*