**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**

**BÀI KIỂM TRA**

|  |  |
| --- | --- |
| Tên môn học | Ngôn ngữ lập trình C++ |
| Họ và tên sinh viên | Nguyễn Tuấn Minh |
| Mã sinh viên | **B24DTCN370** |
| Lớp | **D24TXCN09-B** |

***Một số lưu ý khi làm bài:***

- Điền đầy đủ thông tin họ và tên sinh viên, mã sinh viên;

- Tuân thủ thời gian khi nhận đề bài và submit tệp “.docx” lên hệ thống LMS đúng thời gian; quá hạn trên, bài nộp sẽ không được chấp nhận;

- Đọc kỹ câu hỏi, thực hiện việc code chương trình trên các phần mềm và copy/paste lời giải vào khung đáp án; Đáp án mã chương trình thực thi trên ngôn ngữ C++, bao gồm thư viện, hàm con, chương trình chính; nếu thiếu, sẽ không được tính bài;

**- Lưu ý:** Trong phần code, các Em không sử dụng tiếng Việt để chú thích/comment hay trong các câu lệnh cout. Chỉ sử dụng tiếng Anh.

**Không sử dụng commnent kiểu**: **// Đây là dãy số**

**Không sử dụng tiếng việt trong các câu lệnh: cout << “Bạn hãy nhập dãy số:”**

**Nếu sử dụng, sẽ không được tính bài.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Câu 1:***  Cho mảng A[] gồm n phần tử và mảng B[] gồm m phần tử. Nhiệm vụ của bạn là hợp nhất hai mảng A[] và B[] để được một mảng mới đã được sắp xếp. Ví dụ A[] = {10, 5, 15}, B[] = {20, 3, 2} ta có kết quả là C[] = {2, 3, 5, 10, 15, 20}.  Input:   * Dòng đầu tiên đưa vào số lượng bộ test T. * Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm ba dòng: dòng đầu tiên đưa vào n, m tương ứng với số phần tử của mảng A[] và B[]; dòng tiếp theo là n số A[i] ; dòng cuối cùng là m số B[i]; các số được viết cách nhau một vài khoảng trống. * T, n, m, A[i], B[i] thỏa mãn ràng buộc: 1≤ T ≤100; 1≤ n, m ≤106; -108≤ A[i] ≤108.   Output:   * Đưa ra kết quả mỗi test theo từng dòng.  |  |  | | --- | --- | | Input: | Output: | | 1  3 3  10 5 15  20 3 2 | 2 3 5 10 15 20 |   (a) Viết chương trình theo yêu cầu đề bài   |  | | --- | | ***Đáp án:*** ***#include <iostream>***  ***#include <vector>***  ***#include <algorithm>***  ***using namespace std;***  ***int main()***  ***{***  ***int t;***  ***cin >> t;***  ***while (t--) {***  ***int n, m;***  ***cin >> n >> m;***  ***vector<int> A(n), B(m);***  ***for (int i = 0; i < n; i = i + 1) {***  ***cin >> A[i];***  ***}***  ***for (int i = 0; i < m; i = i + 1) {***  ***cin >> B[i];***  ***}***  ***vector<int> C(n + m);***  ***merge(A.begin(), A.end(), B.begin(), B.end(), C.begin());***  ***sort(C.begin(), C.end());***  ***for (int i = 0; i < n + m; i = i + 1) {***  ***cout << C[i] << " ";***  ***}***  ***cout << endl;***  ***}***  ***return 0;***  ***}*** |   (b) Viết thành hàm xử lý với tham số đầu vào n, m, mảng A[] và mảng B[] đáp ứng yêu cầu của đề bài   |  | | --- | | ***Đáp án:*** | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Câu 2:***  Cho số tự nhiên N và một hoán vị X[] của 1, 2, .., N. Nhiệm vụ của bạn là đưa ra hoán vị trước đó của X[]. Ví dụ N=5, X[] = {1, 2, 3, 4, 5} thì hoán vị trước đó của X[] là {5, 4, 3, 2, 1}.  **Input:**   * Dòng đầu tiên đưa vào số lượng test T. * Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là số N; dòng tiếp theo đưa vào hoán vị X[] của 1, 2, .., N. * T, N, X[] thỏa mãn ràng buộc: 1≤T≤100; 1≤ N≤103. * Input đảm bảo không có trường hợp hoán vị đã cho là đầu tiên (tức là luôn có hoán vị trước nó)   **Output:**   * Đưa ra kết quả mỗi test theo từng dòng.   **Ví dụ:**     |  |  | | --- | --- | | **Input** | **Output** | | 2  5  1  2  3  5  4  5  5  4  3  2  1 | 1  2  3  4  5  5  4  3  1  2 |   (a) Viết chương trình thực hiện yêu cầu trên   |  | | --- | | ***Đáp án: #include <iostream>***  ***#include <vector>***  ***#include <algorithm>***  ***using namespace std;***  ***int main()***  ***{***  ***int a;***  ***cin >> a;***  ***vector<int> X(a);***  ***for (int i = 0; i < a; i = i + 1) {***  ***cin >> X[i];***  ***}***  ***int i = a - 2;***  ***while (i >= 0 && X[i] < X[i + 1]) {***  ***i--;***  ***}***  ***if (i >= 0) {***  ***int j = a - 1;***  ***while (X[j] > X[i]) {***  ***j--;***  ***}***  ***swap(X[i], X[j]);***  ***}***  ***reverse(X.begin() + i + 1, X.end());***  ***for (int i = 0; i < a; i = i + 1) {***  ***cout << X[i] << " ";***  ***}***  ***cout << endl;***  ***return 0;***  ***}*** |   (b) Viết thành hàm với tham số đầu vào là số tự nhiên N và hoán vị X[] của 1, 2, …, N và đáp ứng yêu cầu trên   |  | | --- | | ***Đáp án:*** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Câu 3:***  Cho ma trận A[N][M] bao gồm các số 0 và 1. Ta gọi mỗi miền của ma trận A[][] là nhóm các số 1 được bao quanh bởi các số 0. Hãy tìm số miền của ma trận. Ví dụ số miền của ma trận A[][] là 4.  lAAAAABJRU5ErkJggg==  Input:   * Dòng đầu tiên đưa vào số lượng bộ test T. * Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào N, M là cấp của ma trận A[][]; dòng tiếp theo đưa vào N×M số A[i][j] ; các số được viết cách nhau một vài khoảng trống. * T, M, N, A[i][j] thỏa mãn ràng buộc: 1≤ T ≤100; 1≤ M, N ≤100; 0≤ A[i][j] ≤1.   Output:   * Đưa ra kết quả mỗi test theo từng dòng.  |  |  | | --- | --- | | Input: | Output: | | 2 3 3 1 1 0 0 0 1 1 0 1 4 4 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 | 2  2 |  |  | | --- | | ***Đáp án: #include <iostream>***  ***#include <vector>***  ***using namespace std;***  ***// Helper function to perform DFS on the matrix***  ***void dfs(vector<vector<int>>& A, int i, int j, int N, int M) {***  ***// Check if the current cell is out of bounds or is a 0***  ***if (i < 0 || i >= N || j < 0 || j >= M || A[i][j] == 0) {***  ***return;***  ***}***    ***// Mark the current cell as visited***  ***A[i][j] = 0;***    ***// Recursively call DFS on the neighboring cells***  ***dfs(A, i-1, j, N, M);***  ***dfs(A, i+1, j, N, M);***  ***dfs(A, i, j-1, N, M);***  ***dfs(A, i, j+1, N, M);***  ***}***  ***int countRegions(vector<vector<int>>& A, int N, int M) {***  ***int regions = 0;***    ***// Iterate through the matrix***  ***for (int i = 0; i < N; i++) {***  ***for (int j = 0; j < M; j++) {***  ***// If the current cell is a 1, perform DFS and increment the region count***  ***if (A[i][j] == 1) {***  ***dfs(A, i, j, N, M);***  ***regions++;***  ***}***  ***}***  ***}***    ***return regions;***  ***}***  ***int main() {***  ***int N = 5, M = 5;***  ***vector<vector<int>> A = {***  ***{1, 1, 0, 0, 0},***  ***{0, 1, 0, 0, 1},***  ***{1, 0, 0, 1, 1},***  ***{0, 0, 0, 0, 0},***  ***{1, 0, 1, 0, 1}***  ***};***    ***int regions = countRegions(A, N, M);***    ***cout << "Number of regions: " << regions << endl;***    ***return 0;***  ***}*** | |