Sure bro! Here’s the C++ code for each PBL question in simple and clear format. Each unit has a mini project-style implementation:

✅ UNIT I – Efficient News Feed (Posts, Comments, Likes)

#include <iostream>

#include <unordered\_map>

#include <vector>

Using namespace std;

Struct Post {

Int postId;

String content;

Vector<string> comments;

Int likes;

Post(int id, string txt) {

postId = id;

content = txt;

likes = 0;

}

};

Unordered\_map<int, Post> newsFeed;

Void addPost(int id, string text) {

newsFeed[id] = Post(id, text);

}

Void addComment(int id, string comment) {

newsFeed[id].comments.push\_back(comment);

}

Void likePost(int id) {

newsFeed[id].likes++;

}

Void showFeed() {

For (auto& p : newsFeed) {

Cout << “Post ID: “ << p.second.postId << “, Text: “ << p.second.content << “\n”;

Cout << “Likes: “ << p.second.likes << “\nComments:\n”;

For (string c : p.second.comments)

Cout << “- “ << c << “\n”;

Cout << “---------------------\n”;

}

}

Int main() {

addPost(1, “Hello World!”);

addComment(1, “Nice post”);

likePost(1);

likePost(1);

addPost(2, “Another Post!”);

showFeed();

return 0;

}

✅ UNIT II – E-commerce Product Search & Sort

#include <iostream>

#include <vector>

#include <algorithm>

Using namespace std;

Struct Product {

Int id;

String name;

Int price;

Int rating;

};

Bool sortByPrice(Product a, Product b) {

Return a.price < b.price;

}

Bool sortByRating(Product a, Product b) {

Return a.rating > b.rating;

}

Void showProducts(vector<Product> &products) {

For (auto &p : products)

Cout << p.name << “ | ₹” << p.price << “ | ⭐” << p.rating << endl;

}

Int main() {

Vector<Product> products = {

{1, “Phone”, 12000, 4},

{2, “Laptop”, 45000, 5},

{3, “Headphones”, 2000, 3}

};

Cout << “\nSorted by Price:\n”;

Sort(products.begin(), products.end(), sortByPrice);

showProducts(products);

cout << “\nSorted by Rating:\n”;

sort(products.begin(), products.end(), sortByRating);

showProducts(products);

return 0;

}

✅ UNIT III – Tree Based Social Network Friend Suggestion

#include <iostream>

Using namespace std;

Struct Node {

Int userID;

Node\* left;

Node\* right;

Node(int id) {

userID = id;

left = right = NULL;

}

};

Void inorder(Node\* root) {

If (!root) return;

Inorder(root->left);

Cout << “User ID: “ << root->userID << endl;

Inorder(root->right);

}

Node\* insert(Node\* root, int id) {

If (!root) return new Node(id);

If (id < root->userID)

Root->left = insert(root->left, id);

Else

Root->right = insert(root->right, id);

Return root;

}

Int main() {

Node\* root = NULL;

Root = insert(root, 50);

Insert(root, 30);

Insert(root, 70);

Insert(root, 20);

Insert(root, 40);

Cout << “In-order Friend Ids:\n”;

Inorder(root);

Return 0;

}

✅ UNIT IV – Graph-Based Shortest Path (Dijkstra for Google Maps)

#include <iostream>

#include <vector>

#include <queue>

Using namespace std;

#define INF 999999

Typedef pair<int, int> pii;

Void dijkstra(vector<vector<pii>> &graph, int src, int V) {

Vector<int> dist(V, INF);

Priority\_queue<pii, vector<pii>, greater<pii>> pq;

Dist[src] = 0;

Pq.push({0, src});

While (!pq.empty()) {

Int u = pq.top().second; pq.pop();

For (auto edge : graph[u]) {

Int v = edge.first;

Int weight = edge.second;

If (dist[v] > dist[u] + weight) {

Dist[v] = dist[u] + weight;

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

}

}

}

For (int i = 0; i < V; i++)

Cout << “Distance to “ << i << “ = “ << dist[i] << endl;

}

Int main() {

Int V = 5;

Vector<vector<pii>> graph(V);

// edges: from, to, weight

Graph[0].push\_back({1, 2});

Graph[0].push\_back({2, 4});

Graph[1].push\_back({2, 1});

Graph[1].push\_back({3, 7});

Graph[2].push\_back({4, 3});

Graph[3].push\_back({4, 1});

Dijkstra(graph, 0, V);

Return 0;

}

✅ UNIT V – Hashing in Ticketing System

#include <iostream>

#include <string>

Using namespace std;

Const int SIZE = 10;

Struct Ticket {

Int id;

String user;

};

Ticket\* hashTable[SIZE];

Int hashFunc(int id) {

Return id % SIZE;

}

Void insert(int id, string user) {

Int index = hashFunc(id);

While (hashTable[index] != NULL)

Index = (index + 1) % SIZE;

hashTable[index] = new Ticket{id, user};

}

Void display() {

For (int i = 0; i < SIZE; i++) {

If (hashTable[i])

Cout << “Slot “ << i << “: TicketID=” << hashTable[i]->id << “, User=” << hashTable[i]->user << endl;

Else

Cout << “Slot “ << i << “: Empty\n”;

}

}

Int main() {

Insert(101, “Amit”);

Insert(111, “Pooja”);

Insert(121, “Ravi”);

Insert(131, “Neha”);

Display();

Return 0;

}

Let me know if you want these combined into one file or converted into Python/Java too.