//1.SCALABILITY ISSUE

#include<stdio.h>

#define muser 5000000;

char s[100][100]={"platform crashes,platform run smoothly"};

int usert(int user,int muser)

{

    if(user>muser)

    {

       return s[0];

    }

    return s[1];

}

void main()

{

    int user;

    printf("Enter the user:");

    scanf("%d",&user)

    usert(user,muser)

    printf("%s",s);

}

//2.RECOMMENDATION ALGORITHM FAILURE

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define TOTALPRODUCT 100

#define FAILURE 0.02

void main()

{

    int failure=0;

    srand(time(0));

    for(int i=0;i<TOTALPRODUCT;i++){

    if(rand()/(double)RAND\_MAX<FAILURE){

    failure++;

    }

    }

    printf("Failed recommendation=%d",failure);

}

//5.TECHNICAL DEBT REDUCTION

#include<stdio.h>

#define LINES\_OF\_CODE 1000000

#define TECHNICAL\_DEBT 0.1

#define REDUCTION\_RATE 0.02

#define MITERATION 10

void main()

{

    int total=LINES\_OF\_CODE\*TECHNICAL\_DEBT;

    int iteration=0;

    printf("Total=%d",&total);

    while(total>0&& iteration<MITERATION){

        printf("Iteration %d:Remaining technical debt:%d\n",iteration,total);

    total-=total\*REDUCTION\_RATE;

    iteration++;

    }

    printf("Technical debt :%d",iteration);

}

//6.ORDER FULLFILLMENT OPTIMIZATION

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#define STAGES 5

char s[STAGES][100] = {"Order Placement", "Inventory Allocation", "Packaging", "Shipping", "Delivery"};

int process(int stage) {

    int delay = rand() % 5 + 1;

    printf("\nStage: %s | Delay: %d seconds", s[stage], delay);

    sleep(delay);

    printf(" | Delay complete.\n");

    return delay;

}

void optimize() {

    int totalOptimizedTime = 0;

    printf("\nOptimizing process:\n");

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

        int optimizedDelay = (rand() % 3) + 1;

        printf("Stage: %s | Optimized Time: %d seconds\n", s[i], optimizedDelay);

        totalOptimizedTime += optimizedDelay;

    }

    printf("Total optimized processing time: %d seconds\n", totalOptimizedTime);

}

int main() {

    srand(time(0));

    int totalTime = 0;

    printf("Processing order...\n");

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

        totalTime += process(i);

    }

    printf("\nTotal processing time: %d seconds\n", totalTime);

    optimize();

    return 0;

}

//4:LOGISTICS AND SUPPLY CHAIN OPTIMIZATION

#include <stdio.h>

#include <limits.h>

#define MAX\_WAREHOUSES 10

#define MAX\_LOCATIONS 20

int minDistance(int dist[], int sptSet[], int V) {

    int min = INT\_MAX, minIndex;

    for (int v = 0; v < V; v++) {

        if (sptSet[v] == 0 && dist[v] <= min) {

            min = dist[v], minIndex = v;

        }

    }

    return minIndex;

}

void dijkstra(int graph[MAX\_WAREHOUSES + MAX\_LOCATIONS][MAX\_WAREHOUSES + MAX\_LOCATIONS], int src, int dist[], int V) {

    int sptSet[V];

    for (int i = 0; i < V; i++) {

        dist[i] = INT\_MAX;

        sptSet[i] = 0;

    }

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {

        int u = minDistance(dist, sptSet, V);

        sptSet[u] = 1;

        for (int v = 0; v < V; v++) {

            if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX && dist[u] + graph[u][v] < dist[v]) {

                dist[v] = dist[u] + graph[u][v];

            }

        }

    }

}

void printSolution(int dist[], int src, int V) {

    printf("Shortest paths from warehouse %d:\n", src + 1);

    for (int i = 0; i < V; i++) {

        if (dist[i] == INT\_MAX) {

            printf("Location %d is unreachable\n", i + 1);

        } else {

            printf("Warehouse/Location %d: %d units\n", i + 1, dist[i]);

        }

    }

}

int main() {

    int W, L;

    printf("Enter the number of warehouses: ");

    scanf("%d", &W);

    printf("Enter the number of delivery locations: ");

    scanf("%d", &L);

    int V = W + L;

    int graph[V][V];

    printf("Enter the distance matrix (total %d elements), separated by spaces:\n", V \* V);

    for (int i = 0; i < V; i++) {

        for (int j = 0; j < V; j++) {

            scanf("%d", &graph[i][j]);

        }

    }

    for (int i = 0; i < W; i++) {

        int dist[V];

        dijkstra(graph, i, dist, V);

        printSolution(dist, i, V);

        printf("\n");

    }

    return 0;

}