1) #include <stdio.h>

#define MAX_USERS 500000

void check_scalability(int current_users) {

    if (current_users > MAX_USERS) {

        printf("Platform Crashed! Too many users: %d\n", current_users);

    } else {

        printf("Platform running fine with %d users.\n", current_users);

    }

}

int main() {

    int current_users;

    printf("Enter number of concurrent users: ");

    scanf("%d", &current_users);

    check_scalability(current_users);

    return 0;

}

2) #include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define TOTAL_RECOMMENDATIONS 100

#define FAILURE_PROBABILITY 0.02

int main() {

    int failed = 0;

    srand(time(0));

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

        double random_value = (double)rand() / RAND_MAX;

        if (random_value < FAILURE_PROBABILITY) {

            failed++;

        }

    }

```
    printf("Total Recommendations: %d\n", TOTAL_RECOMMENDATIONS);
    printf("Failed Recommendations: %d\n", failed);
    return 0;
}
```

3)

```
#include <stdio.h>
#define WAREHOUSES 10
#define MAX_CAPACITY 50
#define ITEMS 5
int main() {
    int stock[ITEMS] = {20, 30, 40, 50, 60};
    int warehouses[WAREHOUSES] = {0};
    optimize_inventory(stock, warehouses);
    return 0;
}
void optimize_inventory(int stock[ITEMS], int warehouses[WAREHOUSES]) {
    int total_stock = 0;
    for (int i = 0; i < ITEMS; i++)
        total_stock += stock[i];
    for (int i = 0; i < WAREHOUSES; i++) {
        if (total_stock > MAX_CAPACITY) {
            warehouses[i] = MAX_CAPACITY;
            total_stock -= MAX_CAPACITY;
        } else {
            warehouses[i] = total_stock;
            total_stock = 0;
        }
    }
    printf("Optimal Inventory Allocation:\n");
    for (int i = 0; i < WAREHOUSES; i++) {
        printf("Warehouse %d: %d items\n", i + 1, warehouses[i]);
    }
```

```
}
```

4)

```
#include <stdio.h>
#include <limits.h>
#define N 5
int minDistance(int dist[], int visited[]) {
    int min = INT_MAX, min_index = -1;
    for (int v = 0; v < N; v++) {
        if (!visited[v] && dist[v] <= min) {
            min = dist[v];
            min_index = v;
        }
    }
    return min_index;
}
void dijkstra(int graph[N][N], int src) {
    int dist[N], visited[N];
    for (int i = 0; i < N; i++) {
        dist[i] = INT_MAX;
        visited[i] = 0;
    }
    dist[src] = 0;
    for (int count = 0; count < N - 1; count++) {
        int u = minDistance(dist, visited);
        visited[u] = 1;
        for (int v = 0; v < N; v++) {
            if (!visited[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }
```

```
    printf("Shortest distances from Warehouse %d:\n", src + 1);
    for (int i = 0; i < N; i++) {
        if (i != src)
            printf("To Location %d: %d\n", i + 1, dist[i]);
    }
}
int main() {
    int graph[N][N];
    printf("Enter %d x %d distance matrix (0 for no direct path):\n", N, N);
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            scanf("%d", &graph[i][j]);
    int warehouse;
    printf("Enter warehouse number (1-%d): ", N);
    scanf("%d", &warehouse);
    dijkstra(graph, warehouse - 1);
    return 0;
}
```

5)

```
#include <stdio.h>
#define TOTAL_LINES 1000000
#define INITIAL_DEBT 0.1
int main() {
    double debt = TOTAL_LINES * INITIAL_DEBT;
    double reduction_rate;
    int months;
    printf("Enter monthly reduction percentage (e.g., 5 for 5%%): ");
    scanf("%lf", &reduction_rate);
    printf("Enter number of months for debt reduction: ");
    scanf("%d", &months);
    reduction_rate /= 100;
    for (int i = 1; i <= months; i++) {
```

```
        debt -= debt * reduction_rate;
        printf("Month %d: Remaining Technical Debt = %.2f lines\n", i, debt);
    }
    return 0;
}
```

6)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define STAGES 5
int main() {
    char *stages[STAGES] = {"Order Receipt", "Inventory Allocation", "Packaging", "Shipping",
"Delivery"};
    int delays[STAGES];
    int total_time = 0;
    srand(time(0));
    printf("Order Fulfillment Simulation:\n");
    for (int i = 0; i < STAGES; i++) {
        delays[i] = rand() % 10 + 1;
        total_time += delays[i];
        printf("%d. %s - Time: %d mins\n", i + 1, stages[i], delays[i]);
    }
    printf("\nTotal Order Fulfillment Time: %d mins\n", total_time);
    return 0;
}
```