1.SCALABILITY ISSUE

```
#include <stdio.h>

#include <stdlib.h>


#define MAX_USERS 500000


int main() {

    int currentUsers = 0, choice;


    while (1) {

        printf("\n1. Login\n2. Logout\n3. Exit\nChoice: ");

        scanf("%d", &choice);


        if (choice == 1) {

            if (currentUsers >= MAX_USERS) {

                printf("Platform Crashed! Too many users.\n");

                exit(1);

            }

            currentUsers++;

            printf("User logged in. Active users: %d\n", currentUsers);

        }

        else if (choice == 2) {

            if (currentUsers > 0) currentUsers--;

            printf("User logged out. Active users: %d\n", currentUsers);
        }
        else if (choice == 3) {
            printf("Exiting. Final users: %d\n", currentUsers);
            break;
        }
        else {
            printf("Invalid choice! Try again.\n");
        }
    }

    return 0;
}
```

2.RECOMMENDATION ALGORITHM FAILURE

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define TOTAL_RECOMMENDATIONS 100
#define FAILURE_PROBABILITY 0.02
int main() {
    int failedRecommendations = 0;
    srand(time(NULL));
    for (int i = 0; i < TOTAL_RECOMMENDATIONS; i++)
{
        double randomValue = (double)rand() / RAND_MAX;

        if (randomValue < FAILURE_PROBABILITY)

{

            failedRecommendations++;

        }

    }

    printf("Total recommendations: %d\n", TOTAL_RECOMMENDATIONS);

    printf("Failed recommendations: %d\n", failedRecommendations);

    printf("Success rate: %.2f%%\n", ((TOTAL_RECOMMENDATIONS - failedRecommendations) /
(double)TOTAL_RECOMMENDATIONS) * 100);


    return 0;

}
```

3.INVENTORY OPTIMIZATION

```
#include <stdio.h>

#define MAX_WAREHOUSES 10

#define MAX_PRODUCTS 10

void allocateProducts(int warehouseCapacities[], int productDemands[], int numWarehouses, int
numProducts) {

    int totalDemandMet = 0;

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

        for (int j = 0; j < numWarehouses; j++) {

            if (warehouseCapacities[j] >= productDemands[i]) {

                warehouseCapacities[j] -= productDemands[i];
                totalDemandMet += productDemands[i];
                printf("Allocated product %d (demand: %d) to warehouse %d (remaining capacity: %d)\n",
                       i + 1, productDemands[i], j + 1, warehouseCapacities[j]);
                break; // Move to the next product
            }
        }
    }
    printf("Total demand met: %d\n", totalDemandMet);
}
int main() {
    int warehouseCapacities[MAX_WAREHOUSES];
    int productDemands[MAX_PRODUCTS];
    int numWarehouses, numProducts;
    printf("Enter the number of warehouses (max %d): ", MAX_WAREHOUSES);
    scanf("%d", &numWarehouses);
    printf("Enter the capacities of each warehouse:\n");
    for (int i = 0; i < numWarehouses; i++) {
        printf("Warehouse %d capacity: ", i + 1);
        scanf("%d", &warehouseCapacities[i]);
    }
    printf("Enter the number of products (max %d): ", MAX_PRODUCTS);
    scanf("%d", &numProducts);
    printf("Enter the demands for each product:\n");
    for (int i = 0; i < numProducts; i++) {
        printf("Product %d demand: ", i + 1);
        scanf("%d", &productDemands[i]);
    }
    allocateProducts(warehouseCapacities, productDemands, numWarehouses, numProducts);


    return 0;
}
```