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(“ERROR: Platform Crashed! Maximum user
limit reached (%d users).\n”, MAX_USERS);
                Break;
            }
            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 {
                Printf(“No active users to log out.\n”);
            }
        }
        Else if (choice == 3) {
            Printf("Exiting. Final active users: %d\n", currentUsers);
            Break;
        }
        Else {
            Printf("Invalid choice! Please enter 1, 2, or 3.\n");
        }
    }

    Return 0;
}
```

2. Recommendation Algorithm Failure

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define FAILURE_PROBABILITY 0.02
int main() {
    int totalRecommendations, failedRecommendations = 0;

    printf("Enter the number of product recommendations to
simulate: ");
    scanf("%d", &totalRecommendations);

    srand(time(0));

    for (int i = 0; i < totalRecommendations; i++) {
        double randValue = (double)rand() / RAND_MAX;
        if (randValue < FAILURE_PROBABILITY) {
            failedRecommendations++;
        }
    }

    printf("\nTotal Recommendations: %d\n",
totalRecommendations);
    printf("Failed Recommendations: %d\n",
failedRecommendations);
    printf("Failure Rate: %.2f%%\n", (failedRecommendations *
100.0) / totalRecommendations);

    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;
            }
        }
    }
    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;
}
```