

Interactive C Course



Run

## Output

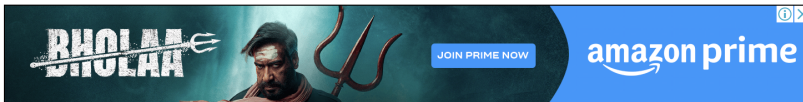
Clear

```

1 #include <stdio.h>
2 #include <stdbool.h>
3 #define NUM_PROCESSES 4
4 #define NUM_RESOURCES 3
5 void checkSafetyState(int claim[][NUM_RESOURCES], int
    allocation[][NUM_RESOURCES], int available[]) {
6     int work[NUM_RESOURCES];
7     bool finish[NUM_PROCESSES];
8     int need[NUM_PROCESSES][NUM_RESOURCES];
9     for (int i = 0; i < NUM_RESOURCES; i++) {
10         work[i] = available[i];
11     }
12     for (int i = 0; i < NUM_PROCESSES; i++) {
13         finish[i] = false;
14     }
15     for (int i = 0; i < NUM_PROCESSES; i++) {
16         for (int j = 0; j < NUM_RESOURCES; j++) {
17             need[i][j] = claim[i][j] - allocation[i][j];
18         }
19     }
20     bool safeState = false;
21     int safeSequence[NUM_PROCESSES];
22     int count = 0;
23     while (count < NUM_PROCESSES) {
24         bool found = false;
25         for (int i = 0; i < NUM_PROCESSES; i++) {
26             if (!finish[i]) {
27                 bool canAllocate = true;

```

```
/tmp/wwz2VZubbx.o
System is in a safe state.
Safe sequence: P0 P1 P2 P3
```



Interactive C Course

main.c



Run

## Output

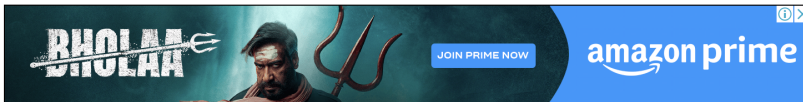
Clear

```

28+         canAllocate = true;
29+         for (int j = 0; j < NUM_RESOURCES; j++) {
30+             if (need[i][j] > work[j]) {
31+                 canAllocate = false;
32+                 break;
33+             }
34+         }
35+         if (canAllocate) {
36+             for (int j = 0; j < NUM_RESOURCES; j++) {
37+                 work[j] += allocation[i][j];
38+             }
39+             safeSequence[count++] = i;
40+             finish[i] = true;
41+             found = true;
42+         }
43+     }
44+     if (!found) {
45+         break;
46+     }
47+ }
48+ if (count == NUM_PROCESSES) {
49+     printf("System is in a safe state.\nSafe sequence: ");
50+     for (int i = 0; i < NUM_PROCESSES; i++) {
51+         printf("P%d ", safeSequence[i]);
52+     }
53+     printf("\n");
54+ } else {

```

```
/tmp/wwz2VZubbx.o
System is in a safe state.
Safe sequence: P0 P1 P2 P3
```



Interactive C Course

main.c

```

41         }
42     }
43 }
44 if (!found) {
45     break;
46 }
47 }
48 if (count == NUM_PROCESSES) {
49     printf("System is in a safe state.\nSafe sequence: ");
50     for (int i = 0; i < NUM_PROCESSES; i++) {
51         printf("%P%d ", safeSequence[i]);
52     }
53     printf("\n");
54 } else {
55     printf("System is in an unsafe state.\n");
56 }
57 }
58 int main() {
59     int claim[NUM_PROCESSES][NUM_RESOURCES] = { {3, 2, 2},{6, 1, 3},{3, 1, 4},
60         , {4, 2, 2} };
61     int allocation[NUM_PROCESSES][NUM_RESOURCES] = {{1, 0, 0},{6, 1, 2},{2, 1
62         , 1},{0, 0, 2}};
63     int available[NUM_RESOURCES] = {9, 3, 6};
64     checkSafetyState(claim, allocation, available);
65     return 0;
66 }
67 }

```

## Output

```
/tmp/wwz2VZubbx.o
System is in a safe state.
Safe sequence: P0 P1 P2 P3
```

Clear