main.c

Run

Output

Clear

```c
#include <stdio.h>
#include <stdbool.h>
#define NUM_PROCESSES 4
#define NUM_RESOURCES 3
void checkSafetyState(int claim[][NUM_RESOURCES], int
    allocation[][NUM_RESOURCES], int available[]) {
    int work[NUM_RESOURCES];
    bool finish[NUM_PROCESSES];
    int need[NUM_PROCESSES][NUM_RESOURCES];
    for (int i = 0; i < NUM_RESOURCES; i++) {
        work[i] = available[i];
    }
    for (int i = 0; i < NUM_PROCESSES; i++) {
        finish[i] = false;
    }
    for (int i = 0; i < NUM_PROCESSES; i++) {
        for (int j = 0; j < NUM_RESOURCES; j++) {
            need[i][j] = claim[i][j] - allocation[i][j];
        }
    }
    bool safeState = false;
    int safeSequence[NUM_PROCESSES];
    int count = 0;
    while (count < NUM_PROCESSES) {
        bool found = false;
        for (int i = 0; i < NUM_PROCESSES; i++) {
            if (!finish[i]) {
                bool canAllocate = true;
```

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

main.c

Run

Output

Clear

```
                bool canAllocate = true;
                for (int j = 0; j < NUM_RESOURCES; j++) {
                    if (need[i][j] > work[j]) {
                        canAllocate = false;
                        break;
                    }
                }
                if (canAllocate) {
                    for (int j = 0; j < NUM_RESOURCES; j++) {
                        work[j] += allocation[i][j];
                    }
                    safeSequence[count++] = i;
                    finish[i] = true;
                    found = true;
                }
            }
        }
        if (!found) {
            break;
        }
    }
    if (count == NUM_PROCESSES) {
        printf("System is in a safe state.\nSafe sequence: ");
        for (int i = 0; i < NUM_PROCESSES; i++) {
            printf("P%d ", safeSequence[i]);
        }
        printf("\n");
    } else {
```

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

Programiz
C Online Compiler

Interactive C Course

main.c

Run

```
                found = true;
            }
        }
    }
    if (!found) {
        break;
    }
}
if (count == NUM_PROCESSES) {
    printf("System is in a safe state.\nSafe sequence: ");
    for (int i = 0; i < NUM_PROCESSES; i++) {
        printf("P%d ", safeSequence[i]);
    }
    printf("\n");
} else {
    printf("System is in an unsafe state.\n");
}
}
int main() {
    int claim[NUM_PROCESSES][NUM_RESOURCES] = { {3, 2, 2},{6, 1, 3},{3, 1, 4}
    , {4, 2, 2} };
    int allocation[NUM_PROCESSES][NUM_RESOURCES] = {{1, 0, 0},{6, 1, 2},{2, 1
    , 1},{0, 0, 2}};
    int available[NUM_RESOURCES] = {9, 3, 6};
    checkSafetyState(claim, allocation, available);
    return 0;
}

```

Output

Clear

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