```c
#include <stdio.h>


int main() {
   int n, m, i, j, k;
   n = 5; // Number of processes
   m = 3; // Number of resources

   int alloc[5][3] = { {0, 1, 0},  // P0
                       {2, 0, 0},  // P1
                       {3, 0, 2},  // P2
                       {2, 1, 1},  // P3
                       {0, 0, 2} }; // P4

   int max[5][3] = { {7, 5, 3},
                     {3, 2, 2},
                     {9, 0, 2},
                     {2, 2, 2},
                     {4, 3, 3} };

   int avail[3] = {3, 3, 2}; // Available resources

   int f[n], ans[n], ind = 0;
   for (k = 0; k < n; k++) {
      f[k] = 0; // Initially, all processes are unfinished
   }

   int need[n][m];
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
         need[i][j] = max[i][j] - alloc[i][j];
      }
```

```
}

printf("Need Matrix:\n");
for (i = 0; i < n; i++) {
   for (j = 0; j < m; j++) {
      printf("%d ", need[i][j]);
   }
   printf("\n");
}

// Banker's Algorithm main logic
int y = 0;
for (k = 0; k < n; k++) {
   for (i = 0; i < n; i++) {
      if (f[i] == 0) {
         int flag = 0;
         for (j = 0; j < m; j++) {
            if (need[i][j] > avail[j]) {
               flag = 1;
               break;
            }
         }

         if (flag == 0) {
            ans[ind++] = i;
            for (y = 0; y < m; y++)
               avail[y] += alloc[i][y];
            f[i] = 1;
         }
      }
   }
```

```
    }

    int flag = 1;
    for (i = 0; i < n; i++) {
        if (f[i] == 0) {
            flag = 0;
            printf("\nSystem is not safe\n");
            break;
        }
    }

    if (flag == 1) {
        printf("\nSystem is in a safe state.\nSafe sequence is: ");
        for (i = 0; i < n - 1; i++)
            printf("P%d -> ", ans[i]);
        printf("P%d\n", ans[n - 1]);
    }

    return 0;
}
```