# Write a c program to stimulate bankers algorithm for the purpose of deadlock avoidance

```
#include <stdio.h>
#include <stdbool.h>

int main() {
   int P, R;
   printf("Processes and resources: ");
scanf("%d %d", &P, &R);

   int alloc[P][R], max[P][R], need[P][R], avail[R], safeSeq[P];
bool finish[P];

   printf("Enter Allocation Matrix:\n");
   for (int i = 0; i < P; i++)
for (int j = 0; j < R; j++)
scanf("%d", &alloc[i][j]);

   printf("Enter Maximum Matrix:\n");
   for (int i = 0; i < P; i++)
for (int j = 0; j < R; j++)
          scanf("%d", &max[i][j]);

   printf("Enter Available Resources:\n");
   for (int i = 0; i < R; i++)
      scanf("%d", &avail[i]);

   for (int i = 0; i < P; i++)
{        finish[i] = false;
for (int j = 0; j < R; j++)
          need[i][j] = max[i][j] - alloc[i][j];
   }

   int count = 0;     while (count
< P) {        bool found = false;
for (int i = 0; i < P; i++) {
if (!finish[i]) {               bool
canRun = true;                for
(int j = 0; j < R; j++)
                if (need[i][j] > avail[j]) canRun =
false;               if (canRun) {
                for (int j = 0; j < R; j++)
                  avail[j] += alloc[i][j];
                safeSeq[count++] = i;
finish[i] = true;
                found = true;
```

```
            }
        }
    }
    if (!found) break;
  }

  if (count == P) {
printf("Safe sequence: ");
for (int i = 0; i < P; i++)
printf("P%d ", safeSeq[i]);
    printf("\n");
  } else {
    printf("System is NOT in a safe state.\n");
  }

  return 0;
}
```

## OUTPUT: