Write a C program to simulate:

a) Bankers' algorithm for the purpose of deadlock avoidance

```
#include <stdio.h>

int main() {
   int n, m, i, j, k;

   printf("Enter number of processes: ");
   scanf("%d", &n);

   printf("Enter number of resource types: ");
   scanf("%d", &m);

   int alloc[n][m], max[n][m], need[n][m], avail[m];
   int finish[n], safeSeq[n];

   printf("Enter allocation matrix:\n");
   for(i = 0; i < n; i++)
      for(j = 0; j < m; j++)
         scanf("%d", &alloc[i][j]);

   printf("Enter maximum matrix:\n");
   for(i = 0; i < n; i++)
      for(j = 0; j < m; j++)
         scanf("%d", &max[i][j]);

   printf("Enter available resources:\n");
   for(i = 0; i < m; i++)
      scanf("%d", &avail[i]);

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

   for(i = 0; i < n; i++)
      finish[i] = 0;

   int count = 0;
   while(count < n) {
      int found = 0;
      for(i = 0; i < n; i++) {
         if(finish[i] == 0) {
            int canRun = 1;
            for(j = 0; j < m; j++) {
               if(need[i][j] > avail[j]) {
```

```
                canRun = 0;
                break;
              }
            }
            if(canRun) {
              for(k = 0; k < m; k++)
                avail[k] += alloc[i][k];
              safeSeq[count++] = i;
              finish[i] = 1;
              found = 1;
            }
          }
        }
        if(!found) {
          printf("System is not in a safe state.\n");
          return 0;
        }
      }

      printf("System is in a safe state.\nSafe sequence: ");
      for(i = 0; i < n; i++)
        printf("P%d ", safeSeq[i]);
      printf("\n");

      return 0;
}
```

Output

```
Enter number of processes: 2
Enter number of resource types: 2
Enter allocation matrix:
321
2
3
6
Enter maximum matrix:
3
3
3
3
Enter available resources:
35
5
System is in a safe state.
Safe sequence: P0 P1


=== Code Execution Successful ===6
```