ASSIGNMENT: 4 (B)

```
#include <stdio.h>
#include <stdbool.h>
bool canAllocate(int need[], int available[], int R)
{
  for (int i = 0; i < R; i++)
  {
     if (need[i] > available[i])
        return false;
  }
  return true;
}
int main()
{
  int P, R;
  printf("Enter number of processes: ");
  scanf("%d", &P);
  printf("Enter number of resource types: ");
  scanf("%d", &R);
  int allocation[P][R], max[P][R], need[P][R], available[R];
  bool finish[P];
  int safeSeq[P];
  printf("Enter allocation matrix (process-wise):\n");
  for (int i = 0; i < P; i++)
  {
     printf("Process %d: ", i);
     for (int j = 0; j < R; j++)
     {
        scanf("%d", &allocation[i][j]);
     }
     finish[i] = false;
  }
  printf("Enter max matrix (process-wise):\n");
  for (int i = 0; i < P; i++)
  {
     printf("Process %d: ", 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", &available[i]);
  }
  // Calculate Need matrix
  for (int i = 0; i < P; i++)
```

```
    {
        for (int j = 0; j < R; j++)
        {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }
    int count = 0;
    int work[R];
    for (int i = 0; i < R; i++)
    {
        work[i] = available[i];
    }
    while (count < P)
    {
        bool found = false;
        for (int i = 0; i < P; i++)
        {
            if (!finish[i] && canAllocate(need[i], work, R))
            {
                for (int j = 0; j < R; j++)
                {
                    work[j] += allocation[i][j];
                }
                safeSeq[count++] = i;
                finish[i] = true;
                found = true;
            }
        }
        if (!found)
        {
            printf("System is NOT in a safe state.\n");
            return 0;
        }
    }
    printf("System is in a safe state.\nSafe sequence is: ");
    for (int i = 0; i < P; i++)
    {
        printf("P%d", safeSeq[i]);
        if (i != P - 1)
            printf(" -> ");
    }
    printf("\n");
    return 0;
}
```

OUTPUT

```
pict@mplab-11:~/Desktop/33164$ gcc -pthread -o bankers_algo bankers_algo.c
pict@mplab-11:~/Desktop/33164$ ./bankers_algo
```

Enter number of processes: 5
Enter number of resource types: 3
Enter allocation matrix (process-wise):
Process 0: 0 0 1
Process 1: 3 0 0
Process 2: 1 0 1
Process 3: 2 3 2
Process 4: 0 0 3
Enter max matrix (process-wise):
Process 0: 7 6 3
Process 1: 3 2 2
Process 2: 8 0 2
Process 3: 2 1 2
Process 4: 5 2 3
Enter available resources:
2 3 2
System is in a safe state.
Safe sequence is: P1 -> P3 -> P4 -> P0 -> P2