```
#include <stdio.h>
#include <stdlib.h>
void final_output(int k[][10], int n, int p)
{
int i, j;
for (i = 0; i < n; i++)
{
printf("\n");
for (j = 0; j < p; j++)
{
printf("%d\t", k[i][j]);
}
}
}
void Banker(int A[][10], int N[][10],
 int M[10][10], int W[1][10], int *n, int *m)
{
int i, j;
printf("\n Enter total number of processes : ");
scanf("%d", n);
printf("\n Enter total number of resources : ");
scanf("%d", m);
for (i = 0; i < *n; i++)
{
printf("\n Process %d\n", i + 1);
for (j = 0; j < *m; j++)
{
printf(" Allocation for resource %d : ", j + 1);
scanf("%d", &A[i][j]);
printf(" Maximum for resource %d : ", j + 1);
scanf("%d", &M[i][j]);
}
}
printf("\n Available resources : \n");
for (i = 0; i < *m; i++)
{
printf(" Resource %d : ", i + 1);
scanf("%d", &W[0][i]);
}
for (i = 0; i < *n; i++)
for (j = 0; j < *m; j++)
N[i][j] = M[i][j] - A[i][j];
printf("\n         Allocation Matrix");
final_output(A, *n, *m);
printf("\n         Maximum Requirement Matrix");
final_output(M, *n, *m);
printf("\n         Need Matrix");
final_output(N, *n, *m);
}
int safety(int A[][10], int N[][10],
int B[1][10], int n, int m, int a[])
{
int i, j, k, x = 0, f1 = 0, f2 = 0;
int F[10], W[1][10];
for (i = 0; i < n; i++)
F[i] = 0;
for (i = 0; i < m; i++)
W[0][i] = B[0][i];
```

```
for (k = 0; k < n; k++)
{
for (i = 0; i < n; i++)
{
if (F[i] == 0)
{
f2 = 0;
for (j = 0; j < m; j++)
{
if (N[i][j] > W[0][j])
f2 = 1;
}
if (f2 == 0 && F[i] == 0)
{
for (j = 0; j < m; j++)
W[0][j] += A[i][j];
F[i] = 1;
f1++;
a[x++] = i;
}
}
}
if (f1 == n)
return 1;
}
return 0;
}
//Resource Request algorithm
void request(int A[10][10], int N[10][10]
, int B[10][10], int pid, int K)
{
int rmat[1][10];
int i;
printf("\n Enter additional request : \n");
for (i = 0; i < K; i++)
{
printf(" Request for resource %d : ", i + 1);
scanf("%d", &rmat[0][i]);
}
for (i = 0; i < K; i++)
if (rmat[0][i] > N[pid][i])
{
printf("\n ******Error encountered******\n");
exit(0);
}
for (i = 0; i < K; i++)
if (rmat[0][i] > B[0][i])
{
printf("\n ******Resources unavailable*****\n");
exit(0);
}
for (i = 0; i < K; i++)
{
B[0][i] -= rmat[0][i];
A[pid][i] += rmat[0][i];
N[pid][i] -= rmat[0][i];
}
}
```

```
int banker(int A[][10], int N[][10],
int W[1][10], int n, int m)
{
int j, i, a[10];
j = safety(A, N, W, n, m, a);
if (j != 0)
{
printf("\n\n");
printf("\n A safety sequence has been "
"detected.\n");
for (i = 0; i < n; i++)
printf(" P%d ", a[i]);
printf("\n");
return 1;
}
else
{
printf("\n Deadlock has occured.\n");
return 0;
}
}
int main()
{
int All[10][10], Max[10][10], Need[10][10]
, W[1][10];
int n, m, pid, c, r;
Banker(All, Need, Max, W, &n, &m);
r = banker(All, Need, W, n, m);
}
```