Name: Shangirne Kharbanda

Registration Number: 20BAI1154

OS LAB-5

Code:

```c
#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]);
}
}
}
//Banker's Algorithm
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("\nAllocation Matrix");
final_output(A, *n, *m);
printf("\nMaximum Requirement Matrix");
final_output(M, *n, *m);
printf("\nNeed Matrix");
final_output(N, *n, *m);
}
//Safety algorithm
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)
```

```
87 if (f1 == n)
88 return 1;
89 }
90 return 0;
91 }
92 //Resource Request algorithm
93 void request(int A[10][10], int N[10][10]
94 , int B[10][10], int pid, int K)
95 {
96 int rmat[1][10];
97 int i;
98 printf("\n Enter additional request : \n");
99 for (i = 0; i < K; i++)
100 {
101 printf(" Request for resource %d : ", i + 1);
102 scanf("%d", &rmat[0][i]);
103 }
104
105 for (i = 0; i < K; i++)
106 if (rmat[0][i] > N[pid][i])
107 {
108 printf("\n ******Error encountered******\n");
109 exit(0);
110 }
111
112 for (i = 0; i < K; i++)
113 if (rmat[0][i] > B[0][i])
114 {
115 printf("\n ******Resources unavailable*****\n");
116 exit(0);
117 }
118
119 for (i = 0; i < K; i++)
120 {
121 B[0][i] -= rmat[0][i];
122 A[pid][i] += rmat[0][i];
123 N[pid][i] -= rmat[0][i];
124 }
125 }
126 int banker(int A[][10], int N[][10],
127 int W[1][10], int n, int m)
128 {
129 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;
printf("\nDEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n");
Banker(All, Need, Max, W, &n, &m);
r = banker(All, Need, W, n, m);
if (r != 0)
{
printf("\n Do you want make an additional"
"request for any of the process ? (1=Yes|0=No)");
scanf("%d", &c);
if (c == 1)
{
printf("\n Enter process number : ");
scanf("%d", &pid);
request(All, Need, W, pid - 1, m);
r = banker(All, Need, W, n, m);
if (r == 0)
{
exit(0);
}
}
}
else
```

```
else
exit(0);
return 0;
}
```

Output:

```
alaric@alaric-virtual-machine:~/Desktop$ gedit bankers.c
alaric@alaric-virtual-machine:~/Desktop$ gcc bankers.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out

DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM

 Enter total number of processes : 5

 Enter total number of resources : 4

 Process 1
 Allocation for resource 1 : 0
 Maximum for resource 1 : 0
 Allocation for resource 2 : 0
 Maximum for resource 2 : 0
 Allocation for resource 3 : 1
 Maximum for resource 3 : 1
 Allocation for resource 4 : 2
 Maximum for resource 4 : 2

 Process 2
 Allocation for resource 1 : 1
 Maximum for resource 1 : 1
 Allocation for resource 2 : 0
 Maximum for resource 2 : 7
 Allocation for resource 3 : 0
 Maximum for resource 3 : 5
 Allocation for resource 4 : 0
 Maximum for resource 4 : 0

 Process 3
 Allocation for resource 1 : 1
 Maximum for resource 1 : 2
 Allocation for resource 2 : 3
 Maximum for resource 2 : 3
 Allocation for resource 3 : 5
 Maximum for resource 3 : 5
 Allocation for resource 4 : 4
 Maximum for resource 4 : 6
```

```
Process 4
Allocation for resource 1 : 0
Maximum for resource 1 : 0
Allocation for resource 2 : 6
Maximum for resource 2 : 6
Allocation for resource 3 : 3
Maximum for resource 3 : 5
Allocation for resource 4 : 2
Maximum for resource 4 : 2

Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 0
Allocation for resource 2 : 0
Maximum for resource 2 : 6
Allocation for resource 3 : 1
Maximum for resource 3 : 5
Allocation for resource 4 : 4
Maximum for resource 4 : 6

Available resources :
Resource 1 : 1
Resource 2 : 5
Resource 3 : 2
Resource 4 : 0
```

```
Allocation Matrix
0       0       1       2
1       0       0       0
1       3       5       4
0       6       3       2
0       0       1       4
Maximum Requirement Matrix
0       0       1       2
1       7       5       0
2       3       5       6
0       6       5       2
0       6       5       6
Need Matrix
0       0       0       0
0       7       5       0
1       0       0       2
0       0       2       0
0       6       4       2


 A safety sequence has been detected.
 P0  P2  P3  P4  P1
```