WEEK 5

Use bankers algo given here to check if the following state is safe/unsafe:

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Is the system in a safe state? If Yes, then what is the safe sequence? What will happen if process P1 requests one additional instance of resource type A and two instances of resource type C?

1. Let *Work* and *Finish* be vectors of length m and n, respectively. Initialize:

   $Work = Available$

   $Finish\ [i] = false$ for i - 1,3, …, n.

2. Find and i such that both:

   (a) $Finish\ [i] = false$

   (b) $Need_i \leq Work$

   If no such i exists, go to step 4.

3. $Work = Work + Allocation_i$

   $Finish[i] = true$

   go to step 2.

4. If $Finish\ [i]$ == true for all i, then the system is in a safe state

Code:

```
#include <stdio.h>

struct file {
   int all[10];
   int max[10];
   int need[10];
   int flag;
};

void main() {
   struct file f[10];
   int fl;
   int i, j, k, p, b, n, r, g, cnt = 0, id, newr;
   int avail[10], seq[10];

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

   printf("Enter number of resources -- ");
   scanf("%d", &r);

   for (i = 0; i < n; i++) {
      printf("Enter details for P%d\n", i);
      printf("Enter allocation -- ");
      for (j = 0; j < r; j++)
```

```
        scanf("%d", &f[i].all[j]);

    printf("Enter Max -- ");
    for (j = 0; j < r; j++)
        scanf("%d", &f[i].max[j]);

    f[i].flag = 0;
}

printf("Enter Available Resources -- ");
for (i = 0; i < r; i++)
    scanf("%d", &avail[i]);

printf("\nEnter New Request Details -- \n");
printf("Enter pid -- ");
scanf("%d", &id);

printf("Enter Request for Resources -- ");
for (i = 0; i < r; i++) {
    scanf("%d", &newr);
    f[id].all[i] += newr;
    avail[i] = avail[i] - newr;
}

for (i = 0; i < n; i++) {
    for (j = 0; j < r; j++) {
```

```
        f[i].need[j] = f[i].max[j] - f[i].all[j];
        if (f[i].need[j] < 0)
            f[i].need[j] = 0;
    }
}

cnt = 0;
fl = 0;
while (cnt != n) {
    g = 0;
    for (j = 0; j < n; j++) {
        if (f[j].flag == 0) {
            b = 0;
            for (p = 0; p < r; p++) {
                if (avail[p] >= f[j].need[p])
                    b = b + 1;
                else
                    b = b - 1;
            }

            if (b == r) {
                printf("\nP%d is visited", j);
                seq[fl++] = j;
                f[j].flag = 1;

                for (k = 0; k < r; k++)
```

```
                avail[k] = avail[k] + f[j].all[k];

            cnt = cnt + 1;
            printf("(");
            for (k = 0; k < r; k++)
                printf("%3d", avail[k]);
            printf(")");
            g = 1;
          }
        }
    }

    if (g == 0) {
        printf("\nREQUEST NOT GRANTED -- DEADLOCK OCCURRED");
        printf("\nSYSTEM IS IN UNSAFE STATE");
        goto y;
    }
}

printf("\nSYSTEM IS IN SAFE STATE");
printf("\nThe Safe Sequence is -- (");
for (i = 0; i < fl; i++)
    printf("P%d ", seq[i]);
printf(")");
```

```
y:
   printf("\n");
}
```

Output:

```
Enter number of processes -- 5
Enter number of resources -- 3
Enter details for P0
Enter allocation -- 0 1 0
Enter Max -- 7 5 3
Enter details for P1
Enter allocation -- 2 0 0
Enter Max -- 3 2 2
Enter details for P2
Enter allocation -- 3 0 2
Enter Max -- 9 0 2
Enter details for P3
Enter allocation -- 2 1 1
Enter Max -- 2 2 2
Enter details for P4
Enter allocation -- 0 0 2
Enter Max -- 4 3 3
Enter Available Resources -- 3 3 2

Enter New Request Details --
Enter pid -- 0
Enter Request for Resources -- 0 0 0

P1 is visited(  5  3  2)
P3 is visited(  7  4  3)
P4 is visited(  7  4  5)
P0 is visited(  7  5  5)
P2 is visited( 10  5  7)
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2 )

Process returned 10 (0xA)   execution time : 67.500 s
Press any key to continue.
```

```
Enter number of processes -- 5
Enter number of resources -- 3
Enter details for P0
Enter allocation -- 0 1 0
Enter Max -- 7 5 3
Enter details for P1
Enter allocation -- 2 0 0
Enter Max -- 3 2 2
Enter details for P2
Enter allocation -- 3 0 2
Enter Max -- 9 0 2
Enter details for P3
Enter allocation -- 2 1 1
Enter Max -- 2 2 2
Enter details for P4
Enter allocation -- 0 0 2
Enter Max -- 4 3 3
Enter Available Resources -- 3 3 2

Enter New Request Details --
Enter pid -- 1
Enter Request for Resources -- 3 0 2

P1 is visited(  5  3  2)
P3 is visited(  7  4  3)
P4 is visited(  7  4  5)
P0 is visited(  7  5  5)
P2 is visited( 10  5  7)
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2 )

Process returned 10 (0xA)   execution time : 65.924 s
Press any key to continue.
```