Week -6

8. Design, develop and implement a C/C++/Java program to implement Banker's algorithm. Assume suitable input required to demonstrate the results.

```
#include <stdio.h>
#include <stdlib.h>
struct process
{
 int alloc[5], max[5], need[5], finished;
} p[10];
int avail[5], req[5], work[5], sseq[10];
int np, nr;
void input()
{
 int i, j, chk = 0;
 printf("Enter The No Of Processes : ");
 scanf("%d", &np);
 printf("Enter The No Of Resources : ");
 scanf("%d", &nr);
 printf("Enter Availability Matrix : \n");
 for (i = 0; i < nr; i++)
 scanf("%d", &avail[i]);
 printf("Enter Allocated Matrix : \n");
 for (i = 0; i < np; i++)
 for (j = 0; j < nr; j++)
 scanf("%d", &p[i].alloc[j]);
 printf("Enter Max Matrix : \n");
 for (i = 0; i < np; i++)
 for (j = 0; j < nr; j++)
 {
 scanf("%d", &p[i].max[j]);
 p[i].need[j] = p[i].max[j] - p[i].alloc[j];
```

```
 if (p[i].need[j] < 0)
 chk = 1;
 }
 if (chk)
 printf("Allocation must be Less than Max\n");
}
int safe()
{
 int flag, sp = 0, i, j;
 for (i = 0; i < nr; i++)
 work[i] = avail[i];
 for (i = 0; i < np; i++)
 p[i].finished = 0;
 while (sp != np)
 {
 flag = 0;
 for (i = 0; i < np; i++)
 {
 if (p[i].finished)
 continue;

int less = 1;
 for (j = 0; j < nr; j++)
 if (p[i].need[j] > work[j])
 less = 0;
 if (less)
 {
 p[i].finished = 1;
 flag = 1;
 sseq[sp++] = i;
 for (j = 0; j < nr; j++)
```

```
work[j] += p[i].alloc[j];
}
}
if (!flag)
{
printf("No Safe Sequence\n");
return 0;
}
}
printf("Safe Sequence \n");
for (i = 0; i < np; i++)
printf("P%d ", sseq[i]);
printf("\n");
return 1;
}
void newReq()
{
int pid, i, j, chk1 = 0, chk2 = 0;
printf("Enter Process ID : ");
scanf("%d", &pid);
printf("Enter Request Matrix : \n");
for (j = 0; j < nr; j++)
{
scanf("%d", &req[j]);
if (req[j] > p[pid].need[j])
chk1 = 1;
if (req[j] > avail[j])
chk2 = 1;
}
if (chk1)
{
```

```
 printf("Process Exceeds Max Need\n");
 return;
 }
 if (chk2)
 {
 printf("Lack Of Resources\n");
 return;
 }
 for (j = 0; j < nr; j++)
{
 avail[j] -= req[j];
 p[pid].alloc[j] += req[j];
 p[pid].need[j] -= req[j];
 }
 if (!safe())
 {
 for (j = 0; j < nr; j++)
 {
 avail[j] += req[j];
 p[pid].alloc[j] -= req[j];
 p[pid].need[j] += req[j];
 }
 }
 else
 printf("Request Committed\n");
}
void display()
{
 int i, j;
 printf("Number of Process : %d\n", np);
 printf("Number of Resources : %d\n", nr);
```

```
 printf("PID\tMax\tAllocated\tNeed\n");
 for (i = 0; i < np; i++)
 {
 printf("P%d\t", i);
 for (j = 0; j < nr; j++)
 printf("%d ", p[i].max[j]);
 printf("\t");
 for (j = 0; j < nr; j++)
 printf("%d ", p[i].alloc[j]);
 printf("\t");
 for (j = 0; j < nr; j++)
 printf("%d ", p[i].need[j]);
 printf("\n");
 }
 printf("Available\n");
 for (i = 0; i < nr; i++)
 printf("%d ", avail[i]);
 printf("\n");
}
void main1()
{
 int ch;
 for (;;)
 {
 printf("1)Input 2)NewRequest 3)Safe 4)Display 5)Exit\n");
 printf("Enter Choice : ");
 scanf("%d", &ch);
 switch (ch)
{
 case 1:
 input();
```

```
break;
case 2:
newReq();
break;
case 3:
safe();
break;
case 4:
display();
break;
case 5:
exit(0);
}
}
}
/*OUTPUT
1)Input 2)NewRequest 3)Safe 4)Display 5)Exit
Enter Choice : 1
Enter The No Of Processes : 5
Enter The No Of Resources : 3
Enter Availability Matrix :
3 3 2
Enter Allocated Matrix :
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter Max Matrix :
7 5 3
3 2 2
```

9 0 2

2 2 2

4 3 3

1)Input 2)NewRequest 3)Safe 4)Display 5)Exit

Enter Choice : 4

Number of Process : 5

Number of Resources : 3

PID Max Allocated Need

P0 7 5 3 0 1 0 7 4 3

P1 3 2 2 2 0 0 1 2 2

P2 9 0 2 3 0 2 6 0 0

P3 2 2 2 2 1 1 0 1 1

P4 4 3 3 0 0 2 4 3 1

Available

3 3 2

1)Input 2)NewRequest 3)Safe 4)Display 5)Exit

Enter Choice : 3

Safe Sequence

P1 P3 P4 P0 P2

1)Input 2)NewRequest 3)Safe 4)Display 5)Exit

Enter Choice : 2

Enter Process ID : 1

Enter Request Matrix :

1 0 2

Safe Sequence

P1 P3 P4 P0 P2

Request Committed

1)Input 2)NewRequest 3)Safe 4)Display 5)Exit

Enter Choice : 4

Number of Process : 5

Number of Resources : 3

PID Max Allocated Need

P0 7 5 3 0 1 0 7 4 3

P1 3 2 2 3 0 2 0 2 0

P2 9 0 2 3 0 2 6 0 0

P3 2 2 2 2 1 1 0 1 1

P4 4 3 3 0 0 2 4 3 1

Available

2 3 0

1)Input 2)NewRequest 3)Safe 4)Display 5)Exit

Enter Choice : 2

Enter Process ID : 4

Enter Request Matrix :

3 3 0

Lack Of Resources

1)Input 2)NewRequest 3)Safe 4)Display 5)Exit

Enter Choice : 5

*/