

Name : Anupam Kunwar
Reg : 19BCE1369
Faculty : Amrit Pal

Lab-Fat CSE2005 (L15-L16)

Q1.

Consider last two digits of your enrolment ID as variables **p** and **q**.

For example if your enrolment ID is 20BCE1123 then p=2, q=3

Assume that there are two type of resources R1 and R2.

How you will ensure that there will be no more than **p** processes which are accessing resources R1 and no more than **q** processes which are accessing resource R2.

Solution :

```
#include <iostream>
using namespace std;
// Function to find the need of each process
void calculateNeed(int **need, int **maxm, int **allot, int P, int R)
{
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            need[i][j] = maxm[i][j] - allot[i][j];
}
// Function to find the system is in safe state or not
bool isSafe(int avail[], int **maxm, int **allot, int P, int R)
{
    int i, j;
    int **need;
    need = new int *[P];
    for (i = 0; i < P; i++)
        need[i] = new int[R];
    // Function to calculate need matrix
    calculateNeed(need, maxm, allot, P, R);
    bool finish[P] = {0};
    int safeSeq[P];
    int work[R];
    for (int i = 0; i < R; i++)
        work[i] = avail[i];
    int count = 0;
    while (count < P)
    {
        bool found = false;
        for (int p = 0; p < P; p++)
        {
            if (finish[p] == 0)
            {
                int j;
```

```

for (j = 0; j < R; j++)
if (need[p][j] > work[j])
break;if (j == R)
{
for (int k = 0; k < R; k++)
work[k] += allot[p][k];
safeSeq[count++] = p;
finish[p] = 1;
found = true;
}
}
}
if (found == false)
{
return false;
}
}
return true;
}
// Driver code
int main()
{
int P, R, i, j, proc_num;
cout << "Enter number of processes : ";
cin >> P;
cout << "Enter number of resources : ";
cin >> R;
// Available instances of resources
cout << "\nEnter available instance of resources\n";
int avail[R];
for (i = 0; i < R; i++)
{
cout << "Available instance of resource "<<i+1<<" = ";
cin >> avail[i];
}
int **maxm;
maxm = new int *[P];
for (i = 0; i < P; i++)
maxm[i] = new int[R];
cout << "\nEnter maximum instances per resource that can be used \n";
for (i = 0; i < P; i++)
{
for (j = 0; j < R; j++)
{
cout << "Process "<<i+1<<" maximum usage of resource " << j+1 <<" = ";
cin >> maxm[i][j];
}
}
} // Resources allocated to processes
int **allot;

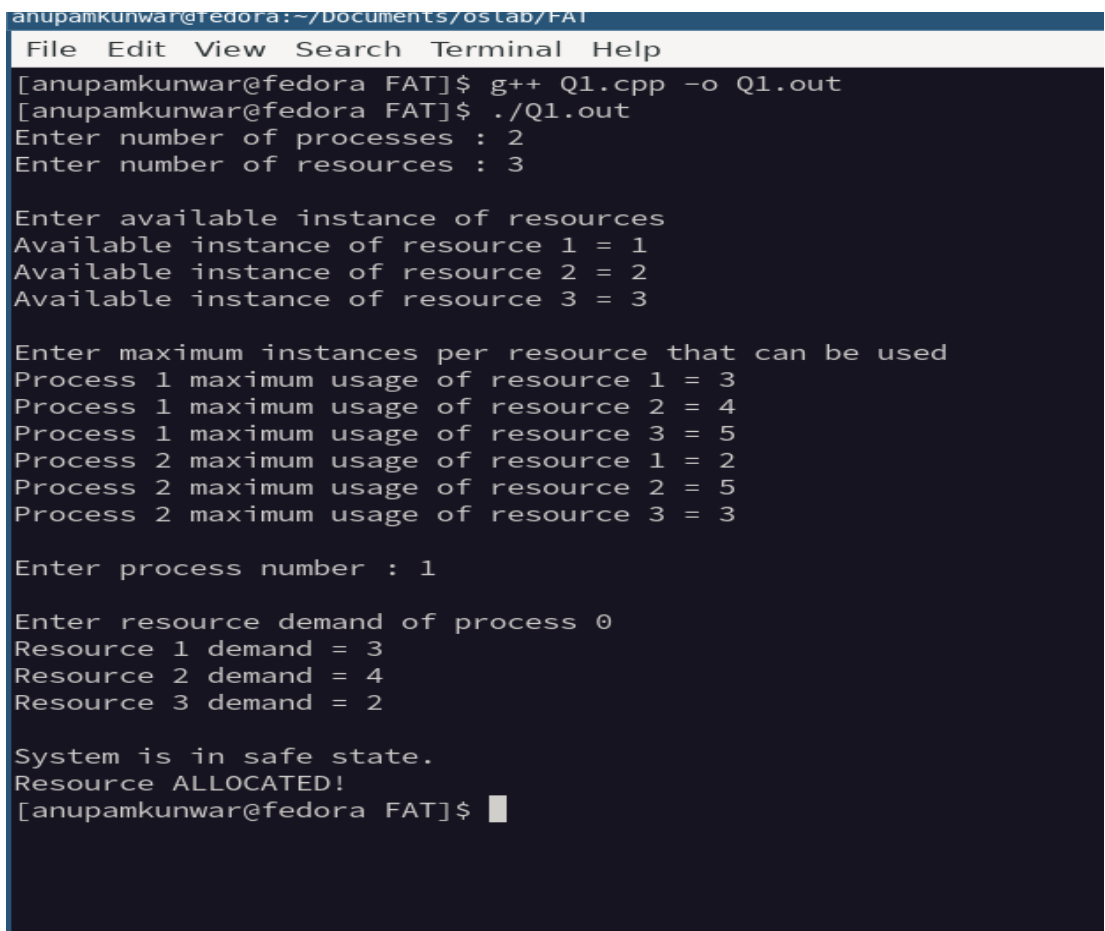
```

```

allot = new int *[P];
for (i = 0; i < P; i++)
allot[i] = new int[R];
cout << "\nEnter process number : ";
cin >> proc_num;
proc_num--;
cout << "\nEnter resource demand of process "<<proc_num<<endl;
for (j = 0; j < R; j++)
{
cout << "Resource "<<j+1<<" demand = ";
cin >> allot[proc_num][j];
}
// Check system is in safe state or not
if(isSafe(avail, maxm, allot, P, R)){
cout << "\nSystem is in safe state.\nResource ALLOCATED!\n";
}
else{
cout << "\nSystem is not in safe state.\nResource NOT ALLOCATED!!\n";
}
return 0;
}

```

Output :



```

anupamkunwar@fedora:~/Documents/oslab/FAT
File Edit View Search Terminal Help
[anupamkunwar@fedora FAT]$ g++ Q1.cpp -o Q1.out
[anupamkunwar@fedora FAT]$ ./Q1.out
Enter number of processes : 2
Enter number of resources : 3

Enter available instance of resources
Available instance of resource 1 = 1
Available instance of resource 2 = 2
Available instance of resource 3 = 3

Enter maximum instances per resource that can be used
Process 1 maximum usage of resource 1 = 3
Process 1 maximum usage of resource 2 = 4
Process 1 maximum usage of resource 3 = 5
Process 2 maximum usage of resource 1 = 2
Process 2 maximum usage of resource 2 = 5
Process 2 maximum usage of resource 3 = 3

Enter process number : 1

Enter resource demand of process 0
Resource 1 demand = 3
Resource 2 demand = 4
Resource 3 demand = 2

System is in safe state.
Resource ALLOCATED!
[anupamkunwar@fedora FAT]$ █

```