Name: Ashmit Thawait

Roll No: 102203790

Group: 2CO-17

Lab Assignment 7

Operating Systems (UCS303)

Ques). Write a program in C/C++/Java to simulate the Banker's algorithm for deadlock avoidance. Consider at least 3 processes in the system, with 4 resource classes having at least one resource instance for each class. Assume the values for Available, Allocation, MAX, and request from a particular process from your side. The program must reflect two cases where a safe sequence exists for one and a safe sequence does not exist for another.

Output:

Not a safe Sequence

```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ touch assignment7.c
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ gcc assignment7.c
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the Max P[0] :: 1
Enter the Max P[0] :: 2
Enter the Max P[0] :: 2
Enter the Max P[0] :: 2
Enter the Max P[1] :: 1
Enter the Max P[1] :: 2
Enter the Max P[1] :: 0
Enter the Max P[1] :: 3
Enter the Max P[2] :: 1
Enter the Max P[2] :: 1
Enter the Max P[2] :: 1
Enter the Max P[2] :: 1

Enter the Allot P[0] :: 0
Enter the Allot P[1] :: 1
Enter the Allot P[2] :: 1
Enter the Allot P[3] :: 0
Enter the Allot P[0] :: 2
Enter the Allot P[1] :: 0
Enter the Allot P[2] :: 1
Enter the Allot P[3] :: 1
Enter the Allot P[0] :: 1
Enter the Allot P[1] :: 0
Enter the Allot P[2] :: 0
Enter the Allot P[3] :: 1

Enter the Resources available :: 1
Enter the Resources available :: 0
Enter the Resources available :: 2
Enter the Resources available :: 0
System is not in safe state
ashmit@ashmit-ubuntu:~/Desktop/ashmit$
```

Safe Sequence:

```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ gcc assignment7.c
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the Max P[0] :: 1
Enter the Max P[0] :: 2
Enter the Max P[0] :: 2
Enter the Max P[0] :: 1
Enter the Max P[1] :: 0
Enter the Max P[1] :: 2
Enter the Max P[1] :: 0
Enter the Max P[1] :: 3
Enter the Max P[2] :: 1
Enter the Max P[2] :: 1
Enter the Max P[2] :: 1
Enter the Max P[2] :: 1

Enter the Allot P[0] :: 1
Enter the Allot P[1] :: 0
Enter the Allot P[2] :: 1
Enter the Allot P[3] :: 1
Enter the Allot P[0] :: 0
Enter the Allot P[1] :: 2
Enter the Allot P[2] :: 0
Enter the Allot P[3] :: 1
Enter the Allot P[0] :: 1
Enter the Allot P[1] :: 1
Enter the Allot P[2] :: 1
Enter the Allot P[3] :: 1

Enter the Resources available :: 1
Enter the Resources available :: 1
Enter the Resources available :: 1
Enter the Resources available :: 1
System is in safe state
 Safe Sequence :2
0
1

Allocating resources to p1...
```

CODE:

```
#include<stdio.h>

#include<stdlib.h>

#define n 3

#define m 4

void calculate_need(int need[n][m], int max[n][m], int allot[n][m])

{

        for(int i=0;i<n;i++)

        {

                for(int j=0;j<m;j++)

                {
```

```
                need[i][j] = max[i][j] - allot[i][j];
                }
        }
}
int is_safe(int process[],int available[],int max[n][m],int allot[n][m])
{
int need[n][m];
calculate_need(need,max,allot);
int finish[n],sequence[n];
for(int i=0;i<n;i++)
{
finish[i]=0;
}
int work[m];
for(int i=0;i<m;i++)
{
work[i]=available[i];
}
int count=0;
while(count<n)
{
int found =0;
for(int p=0;p<n;p++)
{
if(finish[p]==0)
{
int j;
for(j=0;j<m;j++)
{if(need[p][j]>work[j])
```

```
{
break;
}
}
if(j==m)
{
for(int k=0;k<m;k++)
{
work[k]+=allot[p][k];
}
sequence[count++]=p;
finish[p]=1;
found=1;
}
}
}
if(found==0)
{
printf("System is not in safe state\n");
return 0;
}
}
printf("System is in safe state\n Safe Sequence :");
for(int i=0;i<n;i++)
{
printf("%d\n",sequence[i]);
}
printf("\n");
return 1;
```

```
}
int main()
{
int process[n]={0,1,2};
int max[n][m];
for(int i=0;i<n;i++)
{
for(int j=0;j<m;j++)
{
printf("Enter the Max P[%d] :: ",i);
scanf("%d",&max[i][j]);
}
}
int allot[n][m];
printf("\n");
for(int i=0;i<n;i++)
{
for(int j=0;j<m;j++)
{
printf("Enter the Allot P[%d] :: ",j);
scanf("%d",&allot[i][j]);
}
}
int available[m];
printf("\n");
for(int i=0;i<m;i++)
{
printf("Enter the Resources available :: ");
scanf("%d",&available[i]);
```

```
}
int request[m] = {1,0,2,0};
if(is_safe(process,available,max,allot))
{
printf("Allocating resources to p1...\n");
for(int i=0;i<m;i++)
{
available[i]-=request[i];
allot[1][i]+=request[i];
max[1][i]+=request[i];
}
is_safe(process,available,max,allot);
}
return 0;
}
```