CSE2008 (Operating Systems) Lab-6

KHAN MOHD OWAIS RAZA (20BCD7138)

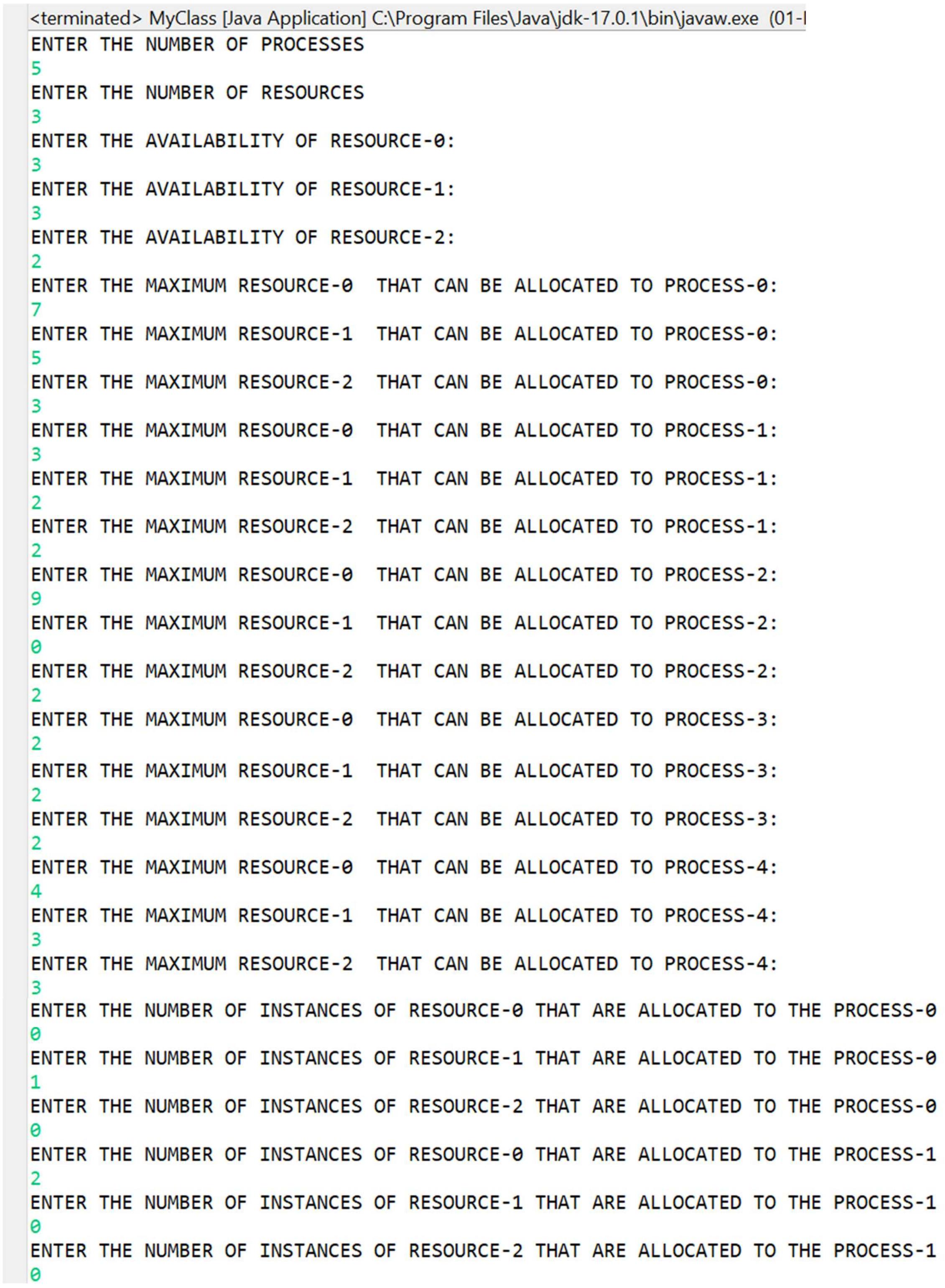
Q.1 Write a program to find safe sequence using Banker's algorithm given number of processes, resources, Allocation, and Max need.

Java Program (Technique-1) :-

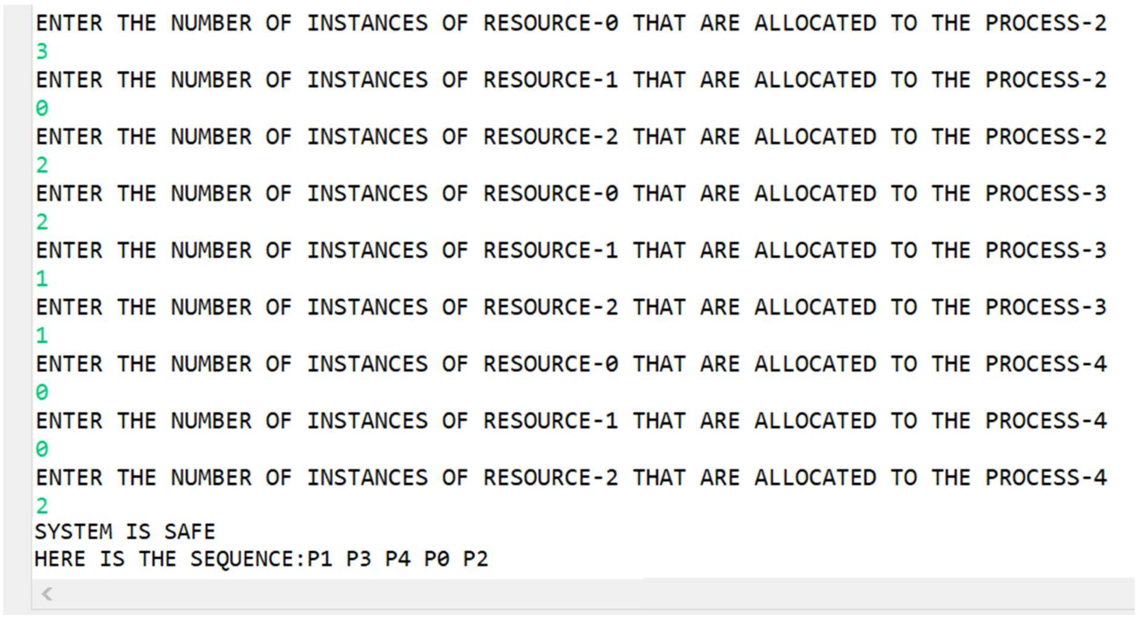
```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138*/
/* Write a program to find safe sequence using
   Banker's algorithm given number of processes,
   resources, allocation, and max need
*/
package owaisraza_20bcd7138;
import java.util.*;
import java.io.*;
import java.util.Scanner;
class MyClass  {
static void FIND_NEED_VALUE(int NEED[][],
                            int MAXIMUM[][],
                            int ALLOCATION[][],
                            int TOTAL_PROCESSES,
                            int TOTAL_RESOURCES){
for (int A = 0 ; A < TOTAL_PROCESSES ; A++){
for (int B = 0 ; B < TOTAL_RESOURCES ; B++){
NEED[A][B] = MAXIMUM[A][B] - ALLOCATION[A][B];
}}
}
static boolean CHECK_SAFE_SYSTEM(int PROCESS[],
                                 int AVAILABLE[],
                                 int MAXIMUM[][],
                                 int ALLOCATION[][],
                                 int TOTAL_PROCESSES,
                                 int TOTAL_RESOURCES) {
int [][]NEED = new int[TOTAL_PROCESSES][TOTAL_RESOURCES];
FIND_NEED_VALUE(NEED, MAXIMUM, ALLOCATION, TOTAL_PROCESSES,
TOTAL_RESOURCES);
boolean []END_PROCESS = new boolean[TOTAL_PROCESSES];
int []SAFE_SEQUENCE = new int[TOTAL_PROCESSES];
int []WORK = new int[TOTAL_RESOURCES];
for (int A = 0; A < TOTAL_RESOURCES ; A++)
WORK[A] = AVAILABLE[A];
int counter = 0;
while (counter < TOTAL_PROCESSES){
boolean foundSafeSystem = false;
for (int X = 0; X < TOTAL_PROCESSES; X++){
```

```
if (END_PROCESS[X] == false){
int B;
for (B = 0; B < TOTAL_RESOURCES; B++)
if (NEED[X][B] > WORK[B])
break;
if (B == TOTAL_RESOURCES){
for (int Y = 0 ; Y < TOTAL_RESOURCES ; Y++)
WORK[Y] += ALLOCATION[X][Y];
SAFE_SEQUENCE[counter++] = X;
END_PROCESS[X] = true;
foundSafeSystem = true;
}}
}
if (foundSafeSystem == false){
System.out.print("THE SYSTEM IS NOT SAFE");
return false;
}}
System.out.print("SYSTEM IS SAFE");
System.out.print("\nHERE IS THE SEQUENCE:");
for (int A = 0; A < TOTAL_PROCESSES ; A++)
System.out.print("N"+SAFE_SEQUENCE[A] + " ");
return true;
}
public static void main(String[] args){
int NUMBER_OF_PROCESSES, NUMBER_OF_RESOURCES;
Scanner sc = new Scanner(System.in);
System.out.println("ENTER THE NUMBER OF PROCESSES");
NUMBER_OF_PROCESSES = sc.nextInt();
System.out.println("ENTER THE NUMBER OF RESOURCES");
NUMBER_OF_RESOURCES = sc.nextInt();
int PROCESS[] = new int[NUMBER_OF_PROCESSES];
for(int A = 0; A < NUMBER_OF_PROCESSES; A++){
PROCESS[A] = A;
}
int AVAILABILITY[] = new int[NUMBER_OF_RESOURCES];
for( int A = 0; A < NUMBER_OF_RESOURCES; A++){
System.out.println("ENTER THE AVAILABILITY OF RESOURCE-"+ A +": ");
AVAILABILITY[A] = sc.nextInt();
}
int MAXIMUM[][] = new int[NUMBER_OF_PROCESSES][NUMBER_OF_RESOURCES];
for( int A = 0; A < NUMBER_OF_PROCESSES; A++){
for( int B = 0; B < NUMBER_OF_RESOURCES; B++){
System.out.println("ENTER THE MAXIMUM RESOURCE-"+ B +" "
                 + " THAT CAN BE ALLOCATED TO PROCESS-"+ A +": ");
MAXIMUM[A][B] = sc.nextInt();
}}
int ALLOCATION[][] = new
int[NUMBER_OF_PROCESSES][NUMBER_OF_RESOURCES];
for( int A = 0; A < NUMBER_OF_PROCESSES; A++){
```

```
for( int B = 0; B < NUMBER_OF_RESOURCES; B++){
System.out.println("ENTER THE NUMBER OF INSTANCES OF RESOURCE-"+ B +
                   " THAT ARE ALLOCATED TO THE PROCESS-"+ A);
ALLOCATION[A][B] = sc.nextInt();
}}
CHECK_SAFE_SYSTEM(PROCESS, AVAILABILITY, MAXIMUM,
                  ALLOCATION, NUMBER_OF_PROCESSES,
                  NUMBER_OF_RESOURCES);
}}
```



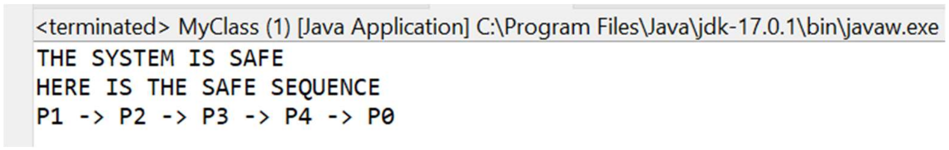
```
<terminated> MyClass [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (01-
ENTER THE NUMBER OF PROCESSES
5
ENTER THE NUMBER OF RESOURCES
3
ENTER THE AVAILABILITY OF RESOURCE-0:
3
ENTER THE AVAILABILITY OF RESOURCE-1:
3
ENTER THE AVAILABILITY OF RESOURCE-2:
2
ENTER THE MAXIMUM RESOURCE-0  THAT CAN BE ALLOCATED TO PROCESS-0:
7
ENTER THE MAXIMUM RESOURCE-1  THAT CAN BE ALLOCATED TO PROCESS-0:
5
ENTER THE MAXIMUM RESOURCE-2  THAT CAN BE ALLOCATED TO PROCESS-0:
3
ENTER THE MAXIMUM RESOURCE-0  THAT CAN BE ALLOCATED TO PROCESS-1:
3
ENTER THE MAXIMUM RESOURCE-1  THAT CAN BE ALLOCATED TO PROCESS-1:
2
ENTER THE MAXIMUM RESOURCE-2  THAT CAN BE ALLOCATED TO PROCESS-1:
2
ENTER THE MAXIMUM RESOURCE-0  THAT CAN BE ALLOCATED TO PROCESS-2:
9
ENTER THE MAXIMUM RESOURCE-1  THAT CAN BE ALLOCATED TO PROCESS-2:
0
ENTER THE MAXIMUM RESOURCE-2  THAT CAN BE ALLOCATED TO PROCESS-2:
2
ENTER THE MAXIMUM RESOURCE-0  THAT CAN BE ALLOCATED TO PROCESS-3:
2
ENTER THE MAXIMUM RESOURCE-1  THAT CAN BE ALLOCATED TO PROCESS-3:
2
ENTER THE MAXIMUM RESOURCE-2  THAT CAN BE ALLOCATED TO PROCESS-3:
2
ENTER THE MAXIMUM RESOURCE-0  THAT CAN BE ALLOCATED TO PROCESS-4:
4
ENTER THE MAXIMUM RESOURCE-1  THAT CAN BE ALLOCATED TO PROCESS-4:
3
ENTER THE MAXIMUM RESOURCE-2  THAT CAN BE ALLOCATED TO PROCESS-4:
3
ENTER THE NUMBER OF INSTANCES OF RESOURCE-0 THAT ARE ALLOCATED TO THE PROCESS-0
0
ENTER THE NUMBER OF INSTANCES OF RESOURCE-1 THAT ARE ALLOCATED TO THE PROCESS-0
1
ENTER THE NUMBER OF INSTANCES OF RESOURCE-2 THAT ARE ALLOCATED TO THE PROCESS-0
0
ENTER THE NUMBER OF INSTANCES OF RESOURCE-0 THAT ARE ALLOCATED TO THE PROCESS-1
2
ENTER THE NUMBER OF INSTANCES OF RESOURCE-1 THAT ARE ALLOCATED TO THE PROCESS-1
0
ENTER THE NUMBER OF INSTANCES OF RESOURCE-2 THAT ARE ALLOCATED TO THE PROCESS-1
0
```


```
ENTER THE NUMBER OF INSTANCES OF RESOURCE-0 THAT ARE ALLOCATED TO THE PROCESS-2
3
ENTER THE NUMBER OF INSTANCES OF RESOURCE-1 THAT ARE ALLOCATED TO THE PROCESS-2
0
ENTER THE NUMBER OF INSTANCES OF RESOURCE-2 THAT ARE ALLOCATED TO THE PROCESS-2
2
ENTER THE NUMBER OF INSTANCES OF RESOURCE-0 THAT ARE ALLOCATED TO THE PROCESS-3
2
ENTER THE NUMBER OF INSTANCES OF RESOURCE-1 THAT ARE ALLOCATED TO THE PROCESS-3
1
ENTER THE NUMBER OF INSTANCES OF RESOURCE-2 THAT ARE ALLOCATED TO THE PROCESS-3
1
ENTER THE NUMBER OF INSTANCES OF RESOURCE-0 THAT ARE ALLOCATED TO THE PROCESS-4
0
ENTER THE NUMBER OF INSTANCES OF RESOURCE-1 THAT ARE ALLOCATED TO THE PROCESS-4
0
ENTER THE NUMBER OF INSTANCES OF RESOURCE-2 THAT ARE ALLOCATED TO THE PROCESS-4
2
SYSTEM IS SAFE
HERE IS THE SEQUENCE:P1 P3 P4 P0 P2
```

Java Program (Technique-2) :-

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138*/
/* CSE2008 (OPERATING SYSTEMS) LAB PRACTICAL-6*/
/* Write a program to find safe sequence using
   Banker’s algorithm given number of processes,
   resources, allocation, and max need
*/
package owaisraza.CSE2008_Lab6;
import java.util.*;
public class MyClass
{
int X = 5; /* Let X = no of resources */
int Y = 3; /* Let Y = no of processes*/
int NEED[][] = new int[X][Y];
int [][]MAXIMUM;
int [][]ALLOCATION;
int []AVAILABLE;
int SAFE_SEQUENCE[] = new int[X];
void INITIALIZE_VALUE(){
ALLOCATION = new int[][] {{ 5, 3, 3 },
                          { 3, 2, 7 },
                          { 3, 0, 2 },
                          { 2, 1, 3 },
                          { 0, 3, 2 }};
MAXIMUM = new int[][] {{ 9, 5, 3 },
                       { 3, 2, 2 },
                       { 7, 0, 2 },
                       { 3, 3, 3 },
                       { 4, 3, 3 }};
AVAILABLE = new int[] { 3, 3, 2 };
}
```

```
void SAFE()
{
int count=0;
boolean VISITED[] = new boolean[X];
for (int A = 0; A < X; A++){
VISITED[A] = false;
}
int WORK[] = new int[Y];
for (int A = 0; A < Y; A++){
WORK[A] = AVAILABLE[A];
}
while (count<X){
boolean flag = false;
for (int A = 0; A < X; A++){
if (VISITED[A] == false){
int B;
for (B = 0; B < Y; B++)
{
if (NEED[A][B] > WORK[B])
break;
}
if (B == Y){
SAFE_SEQUENCE[count++]=A;
VISITED[A]=true;
flag=true;
for (B = 0; B < Y; B++){
WORK[B] = WORK[B]+ALLOCATION[B][B];
}}
}}
if (flag == false){
break;
}}
if (count < X){
System.out.println("THE SYSTEM IS NOT SAFE");
}
else{
System.out.println("THE SYSTEM IS SAFE");
System.out.println("HERE IS THE SAFE SEQUENCE");
for (int A = 0; A < X; A++){
System.out.print("P" + SAFE_SEQUENCE[A]);
if (A != X-1)
System.out.print(" -> ");
}}
}
void CALCULATE_NEED()
{
for (int A = 0; A < X; A++){
for (int B = 0; B < Y; B++){
NEED[A][B] = MAXIMUM[A][B]-ALLOCATION[A][B];
}}
}public static void main(String[] args){
int A, B, C;
MyClass BANKER_ALGORITHM = new MyClass();
BANKER_ALGORITHM.INITIALIZE_VALUE();
BANKER_ALGORITHM.CALCULATE_NEED();
BANKER_ALGORITHM.SAFE();
}}
```

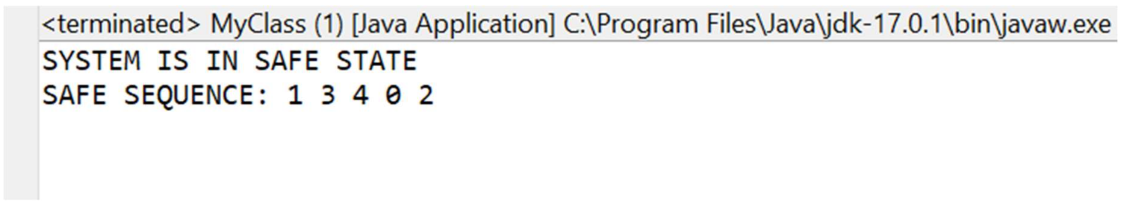


```
<terminated> MyClass (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe
THE SYSTEM IS SAFE
HERE IS THE SAFE SEQUENCE
P1 -> P2 -> P3 -> P4 -> P0
```

Java Program (Technique-3) :-

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138*/
/* CSE2008 (OPERATING SYSTEMS) LAB PRACTICAL-6*/
/* Write a program to find safe sequence using
   Banker's algorithm given number of processes,
   resources, allocation, and max need
*/
package owaisraza.CSE2008_Lab6;
import java.util.*;
class MyClass
{
static int X = 5;
static int Y = 3;
static void CALCULATE_NEED(int NEED[][],
                           int MAXIMUM[][],
                           int ALLOCATION[][]){
for (int A = 0 ; A < X ; A++)
for (int B = 0 ; B < Y ; B++)
NEED[A][B] = MAXIMUM[A][B] - ALLOCATION[A][B];
}
static boolean SAFE(int PROCESSES[],
                    int AVAILABLE[],
                    int MAXIMUM[][],
                    int ALLOCATION[][]){
int [][]NEED = new int[X][Y];
CALCULATE_NEED(NEED, MAXIMUM, ALLOCATION);
boolean []FINISH = new boolean[X];
int []SAFE_SEQUENCE = new int[X];
int []WORK = new int[Y];
for (int A = 0; A < Y; A++)
WORK[A] = AVAILABLE[A];
int count = 0;
while (count < X){
boolean found = false;
```

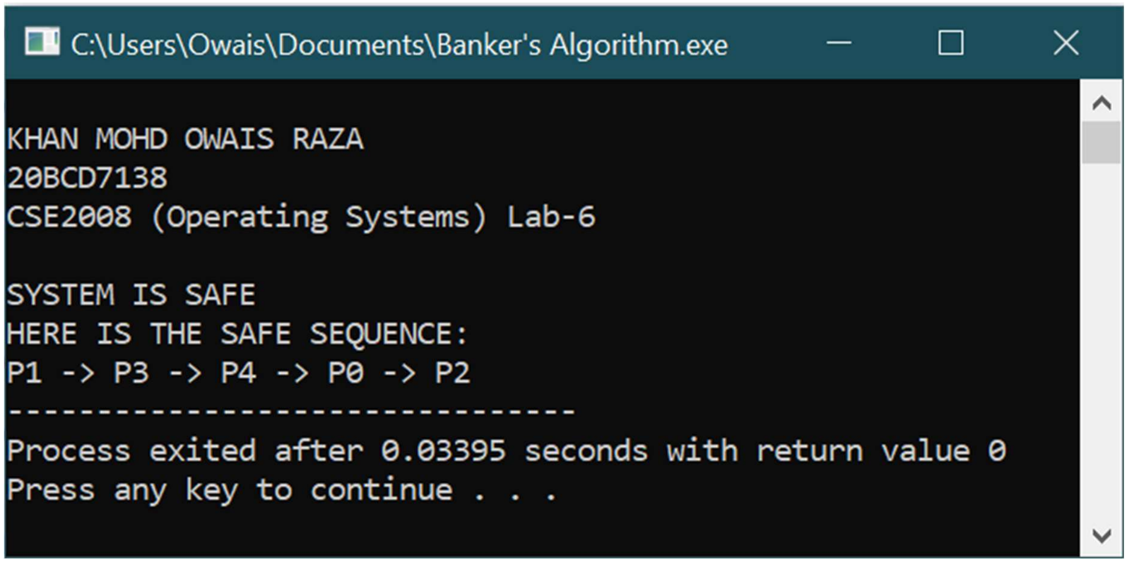
```java
for (int C = 0; C < X; C++){
if (FINISH[C] == false){
int B;
for (B = 0; B < Y; B++)
if (NEED[C][B] > WORK[B])
break;
if (B == Y){
for (int D = 0 ; D < Y ; D++)
WORK[D] += ALLOCATION[C][D];
SAFE_SEQUENCE[count++] = C;
FINISH[C] = true;
found = true;
}}
}
if (found == false){
System.out.print("SYSTEM IS NOT IN SAFE SEQUENCE");
return false;
}}
System.out.print("SYSTEM IS IN SAFE STATE");
System.out.print("\nSAFE SEQUENCE:");
for (int A = 0; A < X ; A++)
System.out.print(SAFE_SEQUENCE[A] + " ");
return true;
}
public static void main(String[] args)
{
int PROCESSES[] = {0, 1, 2, 3, 4};
int AVAILABLE[] = {3, 3, 2};
int MAXIMUM[][] = {{7, 5, 3},
                   {3, 2, 2},
                   {9, 0, 2},
                   {2, 2, 2},
                   {4, 3, 3}};
int ALLOCATION[][] = {{0, 1, 0},
                      {2, 0, 0},
                      {3, 0, 2},
                      {2, 1, 1},
                      {0, 0, 2}};
SAFE(PROCESSES, AVAILABLE, MAXIMUM, ALLOCATION);
}}
```



```
<terminated> MyClass (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe
SYSTEM IS IN SAFE STATE
SAFE SEQUENCE: 1 3 4 0 2
```

C Program :-

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138*/
/* CSE2008 (OPERATING SYSTEMS) LAB PRACTICAL-6*/
/* Write a program to find safe sequence using
   Banker’s algorithm given number of processes,
   resources, allocation, and max need
*/
#include <stdio.h>
int main()
{
int X, Y, A, B, C;
X = 5;
Y = 3;
int ALLOCATION[5][3] = {{0, 1, 0},
                        {2, 0, 0},
                        {3, 0, 2},
                        {2, 1, 1},
                        {0, 0, 2}};
int MAXIMUM[5][3] = {{7, 5, 3},
                     {3, 2, 2},
                     {9, 0, 2},
                     {2, 2, 2},
                     {4, 3, 3}};
int AVAILABILITY[3] = {3, 3, 2};
int f[X], ans[X], ind = 0;
for (C = 0; C < X; C++) {
f[C] = 0;
}
int NEED[X][Y];
for (A = 0; A < X; A++) {
for (B = 0; B < Y; B++)
NEED[A][B] = MAXIMUM[A][B] - ALLOCATION[A][B];
}
int Z = 0;
for (C = 0; C < 5; C++) {
for (A = 0; A < X; A++) {
if (f[A] == 0) {
int flag = 0;
for (B = 0; B < Y; B++) {
if (NEED[A][B] > AVAILABILITY[B]){
flag = 1;
break;
}
}
if (flag == 0) {
ans[ind++] = A;
for (Z = 0; Z < Y; Z++)
AVAILABILITY[Z] += ALLOCATION[A][Z];
f[A] = 1;
}
}
}
}
int flag = 1;
for(int A=0; A<X; A++)
{
if(f[A]==0)
{
flag=0;
printf("\nKHAN MOHD OWAIS RAZA ");
printf("\n20BCD7138");
printf("\nCSE2008 (Operating Systems) Lab-6");
printf("\n ");
printf("SYSTEM IS NOT SAFE");
break;
}
}
if(flag==1){
printf("\nKHAN MOHD OWAIS RAZA ");
printf("\n20BCD7138");
printf("\nCSE2008 (Operating Systems) Lab-6");
printf("\n ");
printf("\nSYSTEM IS SAFE");
printf("\nHERE IS THE SAFE SEQUENCE:\n");
for (A = 0; A < X-1; A++)
printf("P%d => ", ans[A]);
printf("P%d", ans[X-1]);
}
return (0);
}
```



C:\Users\Owais\Documents\Banker's Algorithm.exe

```

KHAN MOHD OWAIS RAZA
20BCD7138
CSE2008 (Operating Systems) Lab-6

SYSTEM IS SAFE
HERE IS THE SAFE SEQUENCE:
P1 -> P3 -> P4 -> P0 -> P2
--------------------------------
Process exited after 0.03395 seconds with return value 0
Press any key to continue . . .
```