5913262 SIWAPORN C.

OPERATING SYSTEM

BANKER'S ALGORITHM

DEADLOCK

WITH FINITE NUMBER OF RESOURCE, THE PROCESS CAN BE AT WAITING STATE AND NEVER ABLE TO CHANGE ITS STATE, THAT IS CALLED “DEADLOCK”

BANKER'S ALGORITHM

- FOR MULTIPLE INSTANCES OF EACH RESOURCE TYPE
- LESS EFFICIENT THAN RESOURCE ALLOCATION SCHEME
- KNOWN FROM BANKING SYSTEM. ENSURING THAT THE BANK WILL NEVER ALLOCATED ITS AVAILABLE CASH IN SUCH WAY THAT IT COULD NO LONGER SATISFIES CUSTOMERS

```
print("Number of process: ")
processNo = readLine()!!.toInt()
print("Number of resource: ")
processResource = readLine()!!.toInt()
print("\n")

input(processNo, processResource, messag
print("\n")

input(processNo, processResource, message
print("\n")

print("Enter Resources: ")
val resourceInput :List<String> = readLine
val resourceInt = ArrayList<Int>()
resourceInput.mapTo(resourceInt) {it.toI
resource = resourceInt
print("---------\n")

print("Allocation:\n")
displayMatrix( input: "allocation")
print("\n")
print("Max:\n")
displayMatrix( input: "max")
```

INPUT PROCESS

```
fun findNeed(resourceNo: Int) {
    for (i :Int in 0 until allocation.size) {
        val tempMatrix = Matrix(resourceNo)
        var temp = ArrayList<Int>(resourceNo)
        for (j :Int in 0 until resourceNo) {
            temp.add(abs( n: max[i].number[j]
            tempMatrix.number = temp
        }
        need.add(tempMatrix)
    }
}

fun findAvailable() {
    for (i :Int in 0 until allocation[0].numb
        var temp:Int = 0
        for (j :Int in 0 until allocation.siz
            temp += allocation[j].number[i]
        }
        available.add(temp) // change here
    }
}

fun findResourceRequire(processNo: Int): Bool
    for (i :Int in 0 until need[processNo].nu
        if (need[processNo].number[i] > vecto
            return false
        }
```

EXECUTING

```
---------Results---------
Allocation Matrix
0 1 1
1 1 1
2 3 4
1 3 4
4 3 4

Max
2 0 3
1 5 6
5 6 7
2 3 4
1 4 2

Sum of resource: [8, 11, 14]
Vector: [2, 6, 7]

Need
2 1 2
```

RESULT

```
1 3 4
1 0 4
0 0 3
9 5 6
9 1 6
1 2 5
14 12 13
```

EXAMPLE CASE

- PROCESSES = 3
- RESOURCES = 3

```
Vector: [13, 9, 2]
Vector: [13, 12, 2]
Vector: [13, 12, 6]
Vector: [14, 12, 6]
Vector: [14, 12, 6]
Vector: [14, 12, 10]
Vector: [14, 12, 10]
Vector: [14, 12, 10]
Vector: [14, 12, 13]

System is in a safe state
Processes: [ p0 p1 p2 ]
Process finished with exi
```

EXAMPLE CASE

- THE SAFE-STATE WITH
- THE PROCESSES OF
- [P0 P1 P2]

THANK YOU!