BACHELOR OF ENGINEERING

(Computer Science Engineering)

(2022-2023)

A PROJECT REPORT ON OPERATING SYSTEM (COM-302)

REPORT ON "HUMAN RESOURCE MANAGEMENT USNIG BANKER'S ALGORITHM"

SUBMITTED BY:

NAME:

SUDANSHI SEHGAL (2021A1R156)

SOUHANI GANDOTRA (2021A1R167)

SIMRAT SINGH (2021A1R170)

ANJALI SALARIA (2021A1R181)

BRANCH: CSE

SEMESTER: 3rd

SECTION: "A3"

PROJECT GROUP: G4

SUBMITTED TO:

Mr. Saurabh Sharma

Ms.Pragti Jamwal

ACKNOWKEDGEMENT

Through this section of our report, we want to present our gratitude towards our institute MIET. From here, we are able to work on projects that require you to analyses and solve a problem that exists in the real world. It is an experience that one can achieve rarely and we are happy and thankful to get that chance here.

We would also like to thank our honourable director Professor Ankur Gupta and our mentors **Assistant Professor Saurabh Sharma**, **Assistant Professor Pragti Jamwal** for being a guiding force throughout this period.

Report on the implementation of the most representative banker's algorithm for deadlock avoidance.

Abstract: This report has described the successful design and implementation of the most representative banker's algorithm for deadlock avoidance. This program is written in C language. Manufacturing systems researchers have dismissed Banker's algorithm as being too conservative for deadlock avoidance in contemporary flexibly automated, discrete-part manufacturing systems. In this report, we provide a modified Banker's logic for the FMS context, and show that the resulting implementations compare favourably in terms of operational flexibility with modern deadlock avoidance policies developed specifically for manufacturing.

PROBLEM STATEMENT:

To implement the most representative banker's algorithm for deadlock avoidance.

INTRODUCTION:

What is Deadlock?

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

The Conditions for Deadlock:

Deadlock can arise if the following four conditions hold simultaneously (Necessary Conditions)

1. Mutual Exclusion:

Only one process at a time may use a shared resource (i.e., critical section).

2. Hold and wait:

A process may hold allocated resources while awaiting assignment of others.

3. No pre-emption:

A resource can be released only voluntarily by the process holding it, after that process holding it, after that process has completed its task.

4. Circular Wait:

A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain.{Deadlock occurs if and only if the circular wait condition is unresolvable.The circular wait condition is unresolvable if the first 3 policy condition hold So, the 4 conditions taken together constitute necessary and sufficient conditions for deadlock.

Banker's Algorithm:

Banker's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resource, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it does not allow the request made by the process.

1. **Inputs to Banker's Algorithm:**

• Max need of resources by each process.

• Currently, allocated resources by each process.

• Max free available resources in the system

2. **The request will only be granted under the below condition:**

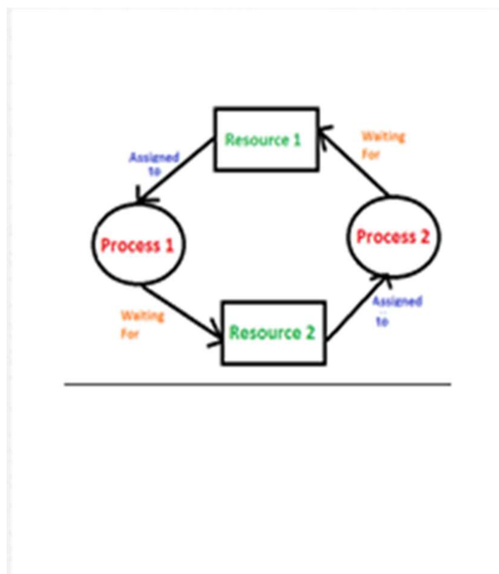
• If the request made by the process is less than equal to max need to that process. Report on the implementation of the most representative banker's algorithm for deadlock avoidance.

• If the request made by the process is less than equal to the freely available resource in the system.

What is Human Resource Management?

Normally, Banker's algorithm is used to manage operating system process & their resources. Same idea is used here. Process is treated as "Projects" and resources are treated as "Project resources". for e.g. In "P1" project, 3 c++, 2 Java, 4 dot net resources are required & so on. This code takes "Claimed resources", "Allocated resources" & "Available resources" as Input and gives proper project completion sequence as output. If unsafe state is occurred then, program will give you suggestion to hire number of employees of particular resources based on logic so that minimum number of resources have to be hired.

FLOWCHART:



TECHNICAL DETAILS:

• **Pseudo Code:**

A safe sequence of processes, or an empty sequence if no such sequence exists.

• **Algorithm:**

1. While there exists a process P in N that has not been terminated do

2. If there exists a resource R in M that P can request and be granted without causing a deadlock then

3. Request R from P

4. Else if P can release a resource R without causing a deadlock, then

5. Release R from P

6. Else

7. Wait

8. End while

9. Return the safe sequence

CODING:

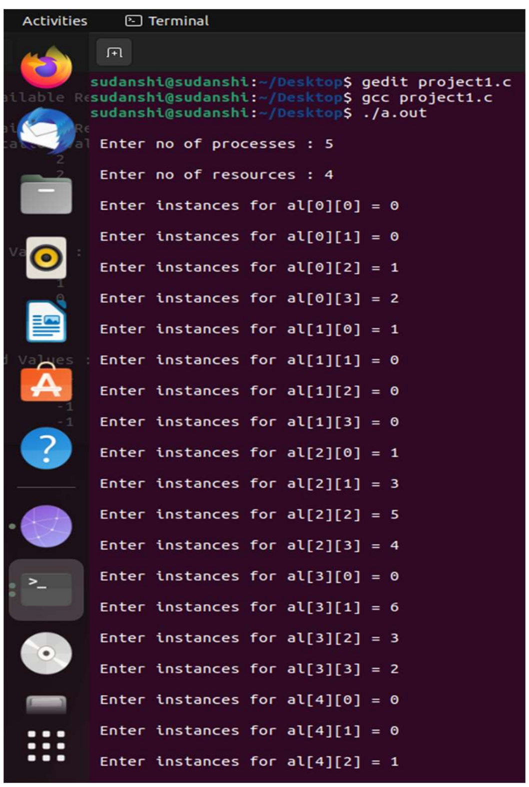
```
#include <stdio.h>

int m, n, i, j, al[10][10], max[10][10], av[10], need[10][10], temp, z, y, p, k;

void main() {

 printf("\n Enter no of processes : "); scanf("%d", &m); // enter numbers of processes

 printf("\n Enter no of resources : "); scanf("%d", &n); // enter numbers of resources

 for (i = 0; i < m; i++) {

 for (j = 0; j < n; j++) {

 printf("\n Enter instances for al[%d][%d] = ", i,j); // al[][] matrix is for allocated instances

 scanf("%d", &al[i][j]);

 // al[i][j]=temp;

        }

}

for (i = 0; i < m; i++) {

for (j = 0; j < n; j++) {

 printf("\n Enter instances for max[%d][%d] = ", i,j); // max[][] matrix is for max instances

 scanf("%d", &max[i][j]);

 }

 }

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

 printf("\n Available Resource for av[%d] = ",i); // av[] matrix is for available instances

 scanf("%d", &av[i]);

 }

 // Print allocation values

 printf("Alocation Values :\n");

 for (i = 0; i < m; i++) {

 for (j = 0; j < n; j++) {
```

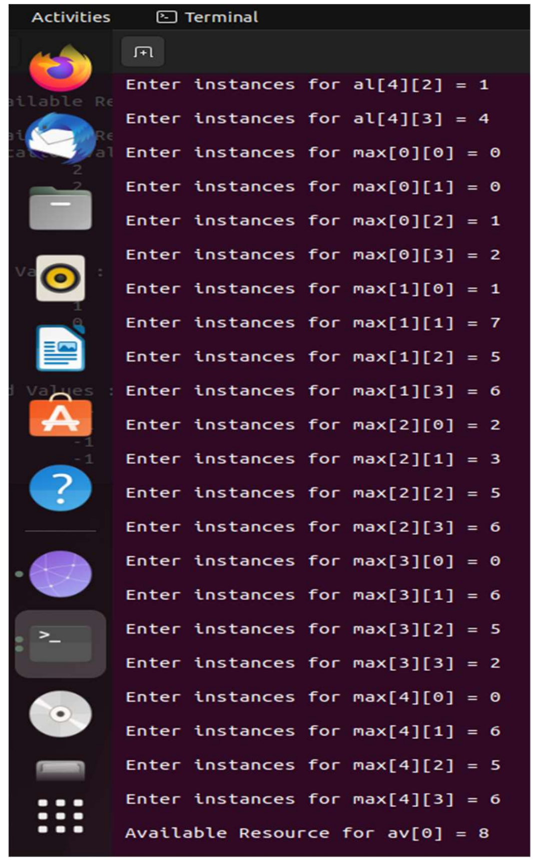
```
printf(" \t %d", al[i][j]); // printing
allocation matrix
}
printf("\n");
 }
 printf("\n\n");
 // Print max values
 printf("Max Values :\n");
 for (i = 0; i < m; i++) {
 for (j = 0; j < n; j++) {
 printf(" \t %d", max[i][j]); // printing max
matrix
}
printf("\n");
 }
 printf("\n\n");
 // Print need values
 printf("Need Values :\n");
 for (i = 0; i < m; i++) {
 for (j = 0; j < n; j++) {
 need[i][j] = max[i][j] - al[i][j]; //
calculating need matrix
 printf("\t %d", need[i][j]);       //  printing
need matrix
}
 printf("\n");
}
 p = 1; // used for terminating while loop
 y = 0;
 while (p != 0) {
for (i = 0; i < m; i++) {
z = 0;
 for (j = 0; j < n; j++) {
 if (need[i][j] <= av[j] &&
(need[i][0] != -1)) { // comparing need
with available instance and
 // checking if the process is done
 // or not
 z++;             // counter if condition TRUE
 }
}
if (z == n) { // if need<=available TRUE
for all resources then condition
// is TRUE
for (k = 0; k < n; k++) {
av[k] += al[i][k]; // new work = work +
allocated
}
 printf("\n Executed process %d", i); //
Print the Process
```

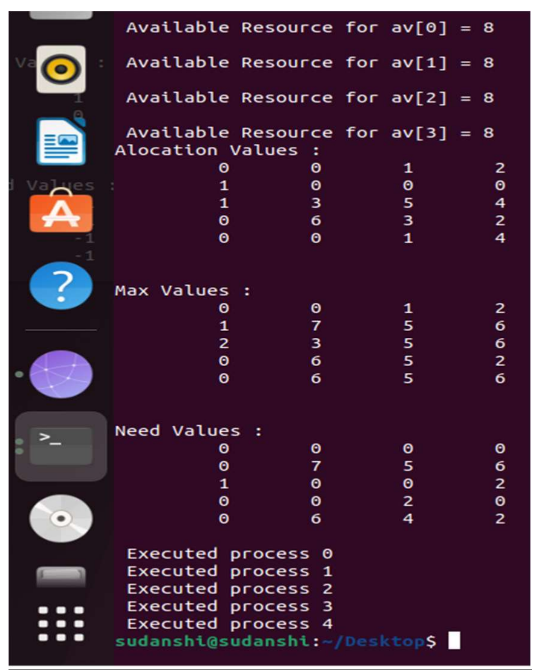
```
 need[i][0] = -1;// assign -1 if Process done
y++;           // cont if process done
 }
 } // end for loop
if (y == m) { // if all done then
p = 0;      // exit while loop
}
  } // end while
printf("\n");
}
```

OUTPUT:

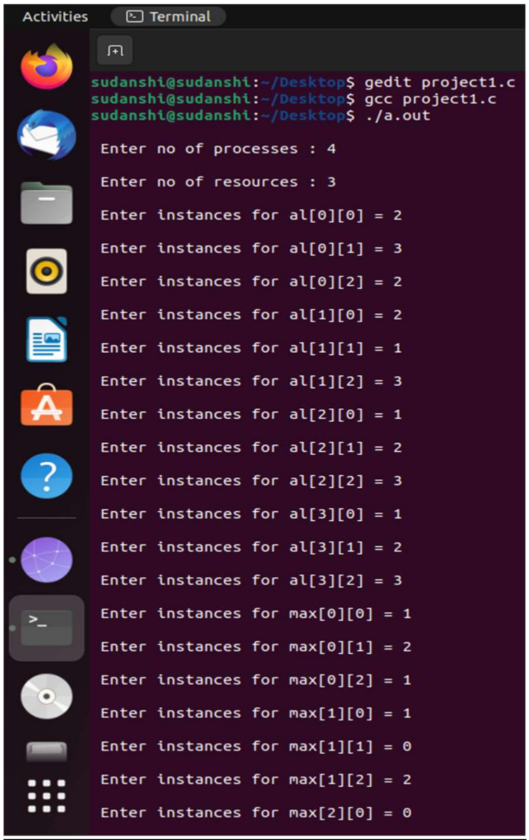
No Deadlock:

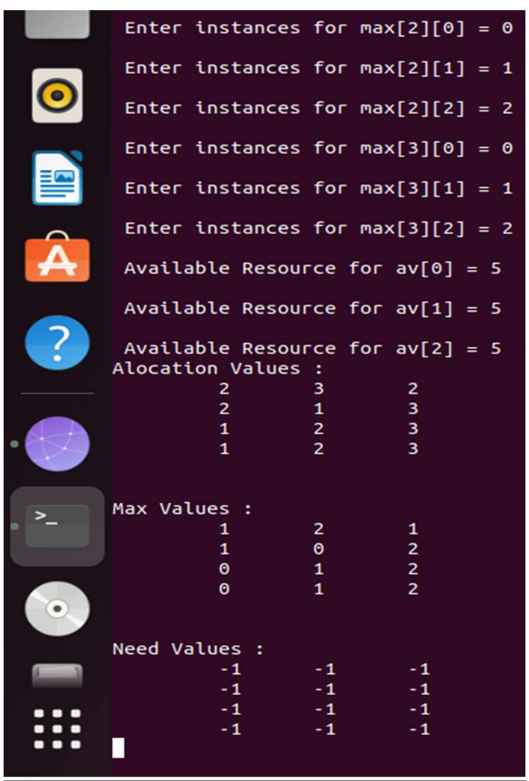






Deadlock:





CONCLUSION AND FUTURE WORK:

Banker's Algorithm helps in detecting and avoiding deadlock and also, helps in managing and controlling process requests of each type of resource within the system. Each process should provide information to the operating system about upcoming resource requests, the number of resources, delays, and about how long the resources will be held by the process before release.

Bankers' algorithm in OS doesn't allow a process to change its maximum need of resources while processing. Another disadvantage of this algorithm is that all the processes within the system must know about the maximum resource needs in advance.

REFERENCES:

1.This paper mainly discusses the application of the algorithm in the course scheduling system for the elective course classroom arrangement https://iopscience.iop.org/article/10.1088/1757-899X/569/5/052020/pdf

2.In this research an approach for Dynamic Banker's algorithm is proposed which allows the number of resources to be changed at runtime that prevents the system to fall in unsafe state. https://ijritcc.org/index.php/ijritcc/article/download/1294/1294/1269

3.In this paper, we studied the principle and data structure of Bankers algorithm, designed the concrete steps of the algorithm.
https://www.scientific.net/AMM.4 22.303