OPERATING SYSTEM

LAB - 2

Faculty- Dr. Abha Trivedi

Name – Fiza Siddiqui Reg. No. - 20BCE10077

Lab Experiment 2

(Bankers Algorithm)

CODE:

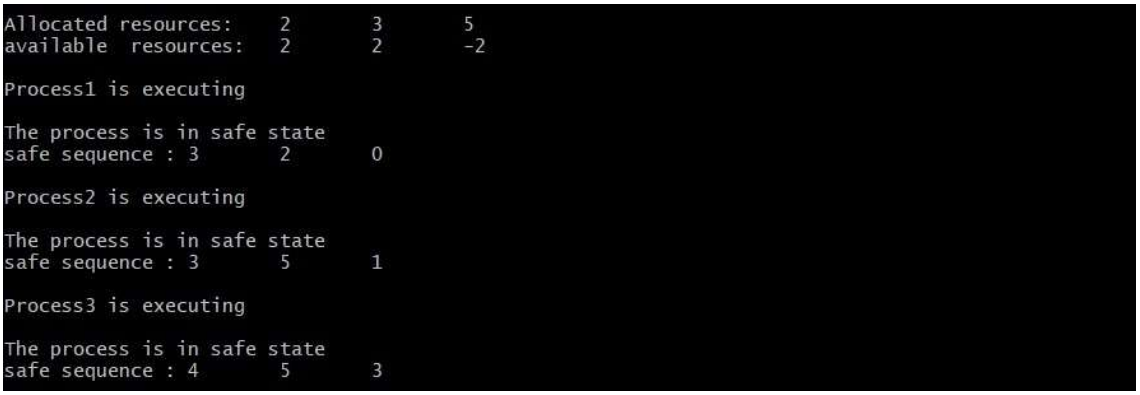
```
#include <stdio.h>

int cur[5][5], claimmax[5][5], avai[5];
int allo[5] = {0, 0, 0, 0, 0};
int max_re[5], running[5], safe = 0;
int count = 0, i, j, exec, re, pro, k = 1;

int main()
{
printf("\nEnter number of processes: ");
     scanf("%d", &pro);

     for (i = 0; i < pro; i++)
{
         running[i] = 1;
         count++;
     }

     printf("\nEnter number of resources: ");
     scanf("%d", &re);

     printf("\nEnter resouce instances  :");
     for (i = 0; i < re; i++)
{
        scanf("%d", &max_re[i]);
     }

   printf("\nEnter Allocated Resource Table:\n");
     for (i = 0; i < pro; i++)
{
        for(j = 0; j < re; j++)
{
   scanf("%d", &cur[i][j]);
         }
     }
```

```
     printf("\nEnter Max resource  Table:\n");
     for (i = 0; i < pro; i++)
{
         for(j = 0; j < re; j++)
{
             scanf("%d", &claimmax[i][j]);
         }
     }

printf("\nThe resource instances  are : ");
     for (i = 0; i < re; i++)
{
        printf("\t%d", max_re[i]);
}

     printf("\nThe Allocated Resource Table:\n");
     for (i = 0; i < pro; i++)
{
        for (j = 0; j < re; j++)
{
             printf("\t%d", cur[i][j]);
         }
printf("\n");
     }

     printf("\nThe Maximum resource Table:\n");
     for (i = 0; i < pro; i++)
{
         for (j = 0; j < re; j++)
{
        printf("\t%d", claimmax[i][j]);
         }
         printf("\n");
     }

     for (i = 0; i < pro; i++)
{
         for (j = 0; j < re; j++)
{
             allo[j] += cur[i][j];
         }
     }

     printf("\nAllocated resources:");
     for (i = 0; i < re; i++)
{
         printf("\t%d", allo[i]);
     }

     for (i = 0; i < re; i++)
{
        avai[i] = max_re[i] - allo[i];
}

     printf("\navailable  resources:");
     for (i = 0; i < re; i++)
```

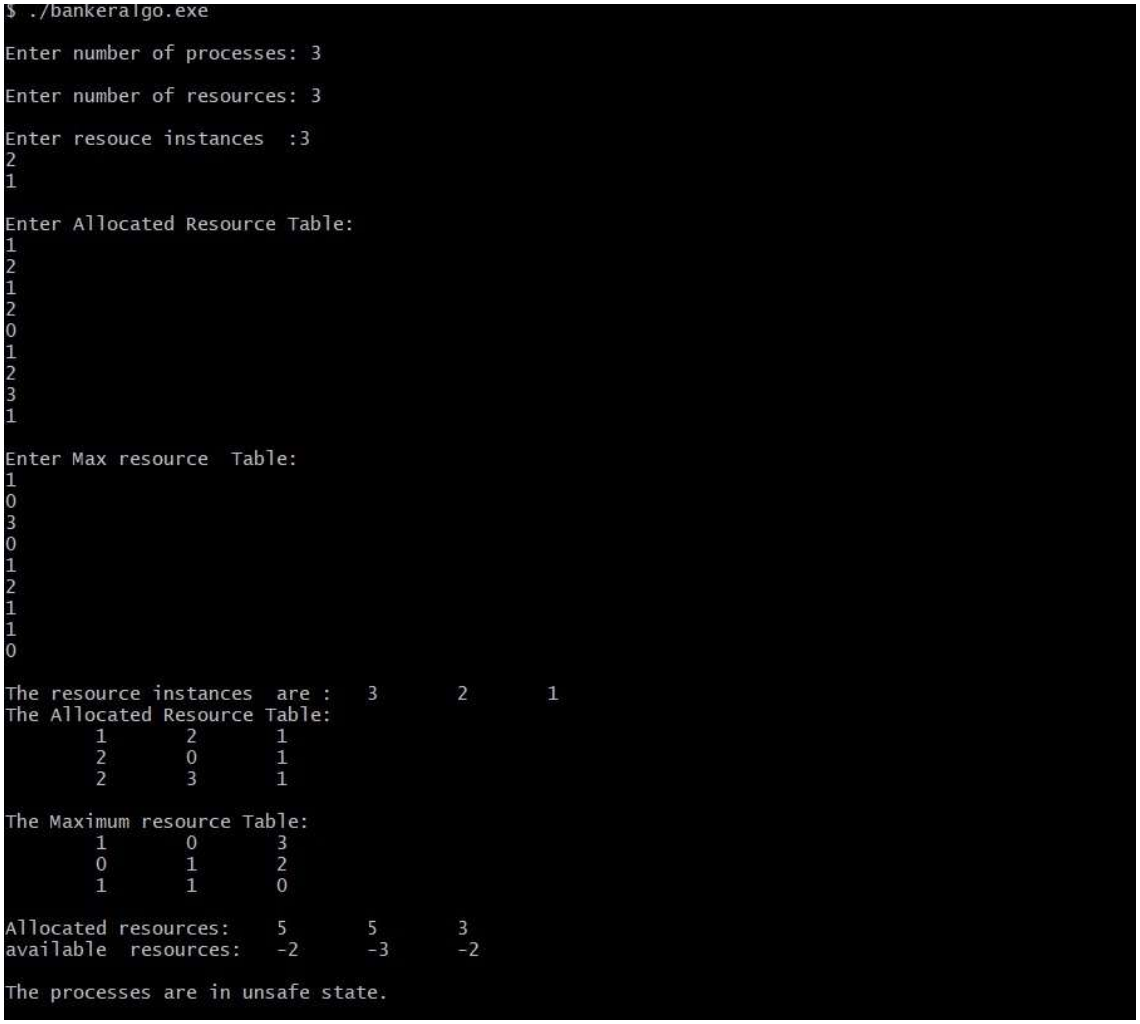
```
{
        printf("\t%d", avai[i]);
    }
    printf("\n");
    //here we check for unsafe and safe state

    while (count != 0)
{
        safe = 0;
        for (i = 0; i < pro; i++)
{
            if (running[i])
{
                exec = 1;
                for (j = 0; j < re; j++)
{
                    if (claimmax[i][j] - cur[i][j] > avai[j])
{
                        exec = 0;
                        break;
                    }
                }
                if (exec)
{
                    printf("\nProcess%d is executing\n", i + 1);
                    running[i] = 0;
                    count--;
                    safe = 1;

                    for (j = 0; j < re; j++)
{
                        avai[j] += cur[i][j];
                    }
                break;
                 }
            }
        }
        if (!safe)
{
            printf("\nThe processes are in unsafe state.\n");
            break;
        }
else
{           printf("\nThe process is in safe state");
            printf("\nsafe sequence :");

            for (i = 0; i < re; i++)
{
                printf("\t%d", avai[i]);
            }

       printf("\n");
        }
    }
    return 0;
}
```

OUTPUT:

Safe Sequence:



Unsafe State:



Explanation:

Lab Experiment

01/12/21

20BCE10077
Fiza Siddiqui
B24

Bankers Algorithm is an algorithm used to avoid deadlocks and allocate resources.

Types of data structures used:

→ Available: used to determine the no. of available resources.
→ Max: Used to determine the max. no. of resources that each process is requesting.
→ Allocation: No. of resources of each kind assigned to each process.
→ Need: Used to determine remaining resources.

Bankers algorithm needs 3 basic things to work:

[MAX]: Maximum resources a process can request.
[AVAILABLE]: Availibility of each resource in system.
[ALLOCATE]: No. of resources each process is currently holding.

Allocation of resources to a process is only possible if when the request made by a process for a resource is less than or equal to actual availibility of resource otherwise processes have to wait and also when the request made by a process At is less than or equal to the maximum resource available else an error occurs.

As explained in the ~~work~~ starting, implementation of Banker's Algorithm is done with the ~~hel~~ help of some basic data structures, that are :

Available : This is an array of length equal to number of resources in system lets say m. It is used to store the available resources in form of instances, lets say k.

Max : This is a nxm matrix where n is number of processes in system & m is ~~no~~ types of resources. This is used to store maximum demand of processes. If in any case Max[i,j] = k (instance) then Process i might be requesting maximum that is equal to k instance of resource j.

Allocation : This is a nxm matrix, where n is no. of processes & m is types of resources in system. It is used to store the ~~maximum~~ allocation of each type of resources made to each process.
Suppose Allo[i,j] ⇒ instance of resource (k) then Process i is currently has allocations of resources j with k instance

The last one is:

Need: This is also an n×m matrix, where n is no. of processes & m is no. of types of resources in system. It is used to store the remaining resources that are yet to be allocated to any process.

Suppose Need[i,] = k (resource instance then Process i has need of k more resources to complete execution.

Need[i,j] = Max[i,j] - Allocation[i,j]

Safe & Unsafe Sequence:

A system is considered in safe state if and only if all its processes have ~~completed~~ completed execution.

We can also say, that safe state is when a system can allocate resources to each process (up to maximum) & still avoid the deadlock.

A system has no way to knowing when a process will terminate, or how many resources have been requested so for, the system will eventually try to reach the specified maximum resources for all processes and soon thereafter.

It is assumed that it ends after that.

It can be said this would be a reasonable explanation / assumption in most cases (from the perview of avoiding deadlock) as the system does not pay particular attention to the execution time of each process. If the process terminates without reaching the maximum resources, the system only simplifies the process.

The safe state is considered a decision maker when processing the ready queue.

Bankers Algorithm determines if a state is safe by trying to find a set of requests made by processes that may allow each processes to acquire maximum resources & then complete the execution. ~~Any state~~

An unsafe state is when no such condition/state/set takes place that is each type of resources are not fully allocated to each process.