

Points:-

1. The program is just an optimisation of Bakery's Safety Algorithm.
2. It targets to reduce the no of cycle taken by safety algorithm to find whether the system is in deadlock or not.

Input Required:-

- All variables name given below are as it is in source code.

Process --> No. of Process

Resources --> No. of Resources

th_id[process] --> Array of Threads (Each Process owns it's separate Thread)

resource_t --> Structure containing information about processes is as follows.

int * FINISH --> Array of integer used to find whether process is entertained or not(0 – false & 1 – true)

int **alloc_resource --> 2-D Array containing information about resources allocated to the process

int **max_resource --> 2-D Array containing information about maximum resources required by process

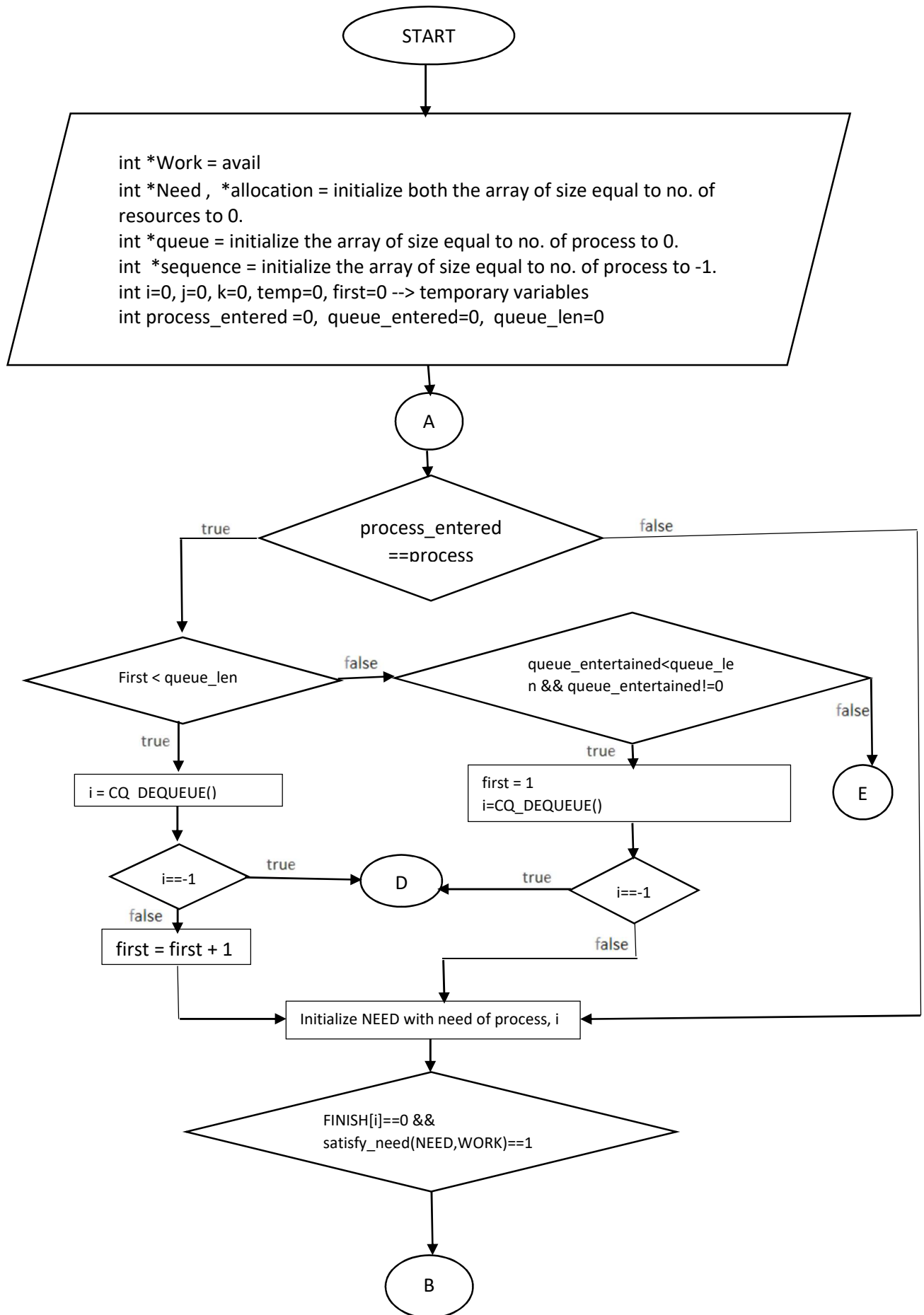
int **req_resource --> 2-D Array containing information about no. of resources requested by process

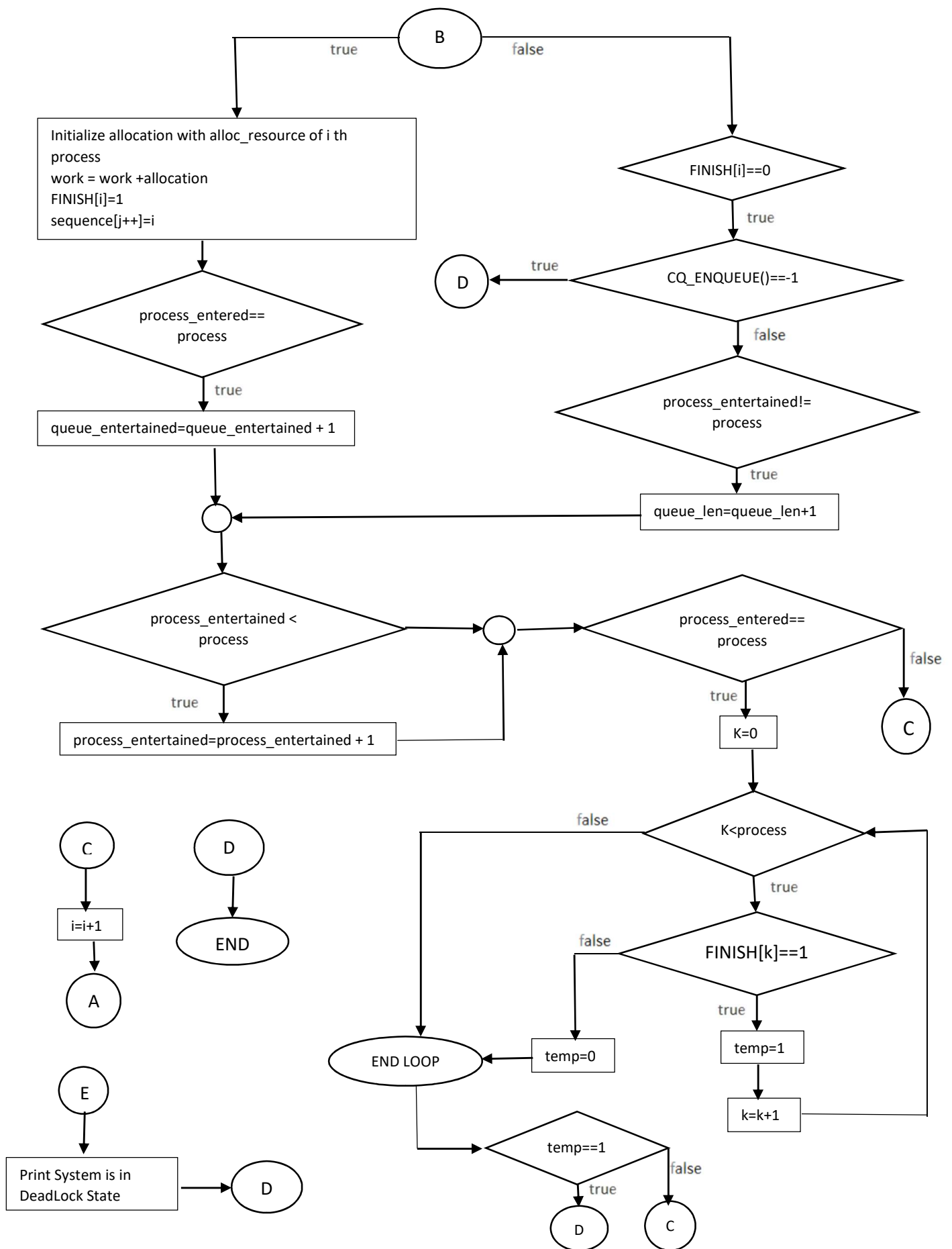
int **need --> 2-D Array giving information about need of each process(It is Calculated using formula $need[i][j] = max[i][j] - alloc[i][j]$)

int *avail --> Array of Available Resources

int *sequence --> Array to store safe sequence

Assuming that we already initialize the resources. Following flow-diagram is of calculating safe-sequence.





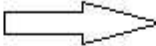
OutPut Explanation :-

Enter The No of Process:-5

Enter The Total Number of Resources:-3

Enter the Allocated resources for process:-

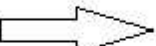
Process	0	1	2
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2



Input for Allocation
Matrix

Enter the Maximum resources for process:-

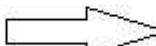
Process	0	1	2
P0	7	5	3
P1	3	2	2
P2	9	0	2
P3	2	2	2
P4	4	3	3



Input for Maximum
Resource Marix

Enter the Request resources for process:-

Process	0	1	2
P0	0	1	0
P1	1	0	0
P2	2	0	0
P3	0	0	0
P4	1	0	1



Input for Request
Resource Matrix

Enter the Available Resources To System:-

AVAIL	0	1	2
3	3	2	

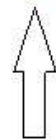


Available Matrix

Each I/P tab Separated

Initially Safe Sequence is:-

Process	Request	Deadlock	State	Available	Initially Safe Sequence Comment
P0	0 1 0	N	Safe	0 1 2	Resource Granted Immediately
P2	2 0 0	N	Unsafe	3 2 2	Sequence After Granting Request is 1 3 4 2 0.
P4	1 0 1	N	Unsafe	3 2 2	Sequence After Granting Request is 3 4 1 2 0.
P1	1 0 0	N	Safe	2 2 2	Resource Granted Immediately



If Deadlock
then Y,
else N



Available Matrix
After Request
Accepted



Comment Regarding
Action Taken by
Sequence Code

Program Explanation:-

1. To optimize number of cycle taken by the safe_sequence code to find whether the system is in deadlock or not I used monitor method. If a process is fail to entertain(i.e. process can not get resources that it want) it will be enqueued in queue.
2. Initially, all process will get chance to check whether it will get entertain or not if entertain then required changes in WORK and AVAIL resources will be done if not it will be enqueued in queue.
3. After all process get chance we will check whether the queue is empty or not, if queue is empty we will break the loop and if not then we will dequeue the process from queue and now will check for it whether it will get entertained or not.
4. Repeat above process till queue won't get empty or if any process from queue won't get entertain then we will say system is in deadlock.

“first” Variable Explanation:-

In program the 'first=1' instruction is written because if any one process get entertained and other doesn't then in next loop we have to set first=1 else loop get terminated.