**Experiment No: 04**

**Experiment Title:** Bankers Algorithm (Deadlock)

**Theory:** The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue.

**Code:**

#include<stdio.h>

int main()

{

int i, j, k, l, q, m = 0, no\_pro = 5, no\_res = 3, flag, need[5][3], avail[3]= {3,3,2}, finish[5]= {0}, seq[5];

int max[5][3] = {{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3}};

int allocate[5][3] = {{0,1,0}, {2,0,0}, {3,0,2}, {2,1,1}, {0,0,2}};

for(i = 0; i < no\_pro; i++)

{

for(j = 0; j < no\_res; j++)

{

need[i][j] = max[i][j] - allocate[i][j];

}

}

for(l = 0; l < no\_pro; l++)

{

for(i = 0; i < no\_pro; i++)

{

if(finish[i] == 0)

{

flag = 0;

for(j = 0; j < no\_res; j++)

{

if(need[i][j] > avail[j])

{

flag = 1;

break;

}

}

if(flag == 0)

{

for(k = 0; k < no\_res; k++)

{

avail[k] = avail[k] + allocate[i][k];

}

finish[i] = 1;

seq[m++] = i;

}

}

}

}

flag = 0;

for(i=0; i<5; i++)

{

if(finish[i]==0)

flag = 1;

}

if(flag==0)

{

printf("Safe\n");

}

else

printf("unsafe\n");

printf("Process are below\n");

for(q = 0; q < no\_pro; q++)

{

printf(" P%d ", seq[q]);

}

}

**Input & Output**:

