Practical 7

Name: Rahul Baser

Roll No.: A4-75

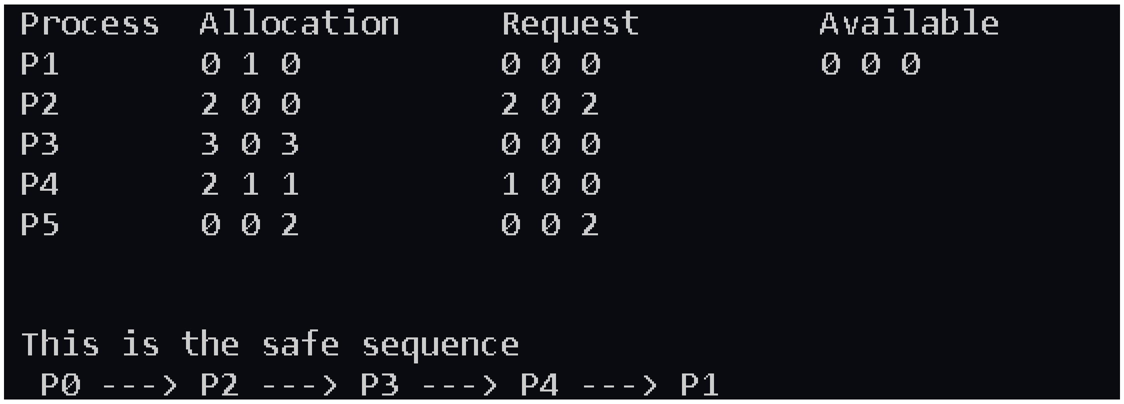
Aim: Write C programs to demonstrate the Banker's Algorithm and recovery processes.

Code:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
	//int pr[5];
	//int av[3];
	//int rq[5][3];
	//int al[5][3];
	int j = 0, k = 0;
	int mx[5][3];

	int pr[5] = {1, 2, 3, 4, 5};

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

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

```
     int av[3] = {0, 0, 0};

     printf("Process  Allocation\tRequest\t\tAvailable\n");
     for(int i = 0; i < 5; i++){
     if(k == 0){
          printf("P%d \t %d %d %d\t\t%d %d %d\t\t%d %d %d\n",
pr[i], al[i][j], al[i][j+1], al[i][j+2], rq[i][j], rq[i][j+1],
rq[i][j+2], av[j], av[j+1], av[j+2]);
     }
     else{
          printf("P%d \t %d %d %d\t\t%d %d %d\n", pr[i],
al[i][j], al[i][j+1], al[i][j+2], rq[i][j], rq[i][j+1],
rq[i][j+2]);
     }
     k++;
     }
     printf("\n\n");

     int n = 5;
     int m = 3;
     int f[n], ans[n], ind = 0;
     for (k = 0; k < n; k++) {
     f[k] = 0;
     }

     int y = 0;
     for (k = 0; k < 5; k++) {
     for (int i = 0; i < n; i++) {
          if (f[i] == 0) {

               int flag = 0;
               for (j = 0; j < m; j++) {
                    if (rq[i][j] > av[j]){
                         flag = 1;
                         break;
                    }
               }

               if (flag == 0) {
                    ans[ind++] = i;
                    for (y = 0; y < m; y++)
                         av[y] += al[i][y];
                    f[i] = 1;
               }
          }
     }
     }

     int flag = 1;

     for(int i=0;i<n;i++)
     {
     if(f[i]==0)
     {
     flag=0;
          printf("This system is not safe!!!\n");
     break;
     }
     }

     if(flag==1)
     {
     printf("This is the safe sequence\n");
     for (int i = 0; i < n - 1; i++)
     printf(" P%d --->", ans[i]);
     printf(" P%d", ans[n - 1]);
     }

     printf("\n\n");

     return 0;
}
```

Output:


```
Process  Allocation     Request         Available
P1       0 1 0          0 0 0           0 0 0
P2       2 0 0          2 0 2
P3       3 0 3          0 0 0
P4       2 1 1          1 0 0
P5       0 0 2          0 0 2


This is the safe sequence
 P0 ---> P2 ---> P3 ---> P4 ---> P1
```