```
Script started on 2019-03-05 23:29:10+0530
praveen@praveen cat Bankers.c
#include<stdio.h>
#include<stdlib.h>
int resource[5];
int allocated[5][5];
int need[5][5];
int max[5][5];
int available[5];
int total[5];
int safety[5];
int overp[5];
int M, N;
int talloc[5][5], tneed[5][5], tavail[5], diff[5];
int c = 0, no = 0, ncheck = 0, ch, dead = 0;
void read()
{
    for(int j = 0; j < M; j++) {
        total[j] = 0;
        diff[j] = 0;
        tavail[j] = 0;
        available[j] = 0;
        overp[j] = 0;
        safety[j] = 0;
        resource[j] = 0;
        for(int i = 0; i < N; i++) {
            allocated[i][j] = 0;
            need[i][j] = 0;
            max[i][j] = 0;
            talloc[i][j] = 0;
            tneed[i][j] = 0;
        }
    }
    c = 0; no = 0; ncheck = 0; dead = 0;
    printf("No. of resources and processes: ");
    scanf("%d %d", &M, &N);
    for(int i = 0; i < M; i++)
    {
        printf("Total no. of resource %d: ", i+1);
        scanf("%d", &resource[i]);
    }
    for(int i = 0; i < N; i++)
    {
        printf("Process %d:\nAllocated: ", i+1);
        for(int j = 0; j < M; j++) {
            scanf("%d", &allocated[i][j]);
            total[j] += allocated[i][j];
        }
        printf("Maximum: ");
        for(int j = 0; j < M; j++) {
            scanf("%d", &max[i][j]);
            need[i][j] = max[i][j] - allocated[i][j];
        }
    }
    for(int i = 0; i < M; i++) {
        available[i] = resource[i] - total[i];
        if(available[i] < 0) {
            printf("Error! Re-enter data!\n");
            for(int j = 0; j < M; j++) available[j] = 0;
            break;
        }
    }
}
```

```
void print()
{
    printf("Data:\nP\tAlloc\t\t Max\t\t  Need\t\t   Available\n \t");
    for(int a = 0; a < 4; a++) {
        for(int b = 0; b < M; b++) {
            printf("R%d ",b+1);
        }
        if(a != 3) printf("\t");
        for(int s = 0; s <= a && a<=2; s++) printf(" ");
    }
    printf("\n \t     \t\t    \t\t      \t\t   ");
    for(int b = 0; b < M; b++) printf("%2d ",available[b]);
    printf("\n");
    for(int i = 0; i < N; i++)
    {
        printf("%d\t", i+1);
        for(int b = 0; b < M; b++) printf("%2d ",allocated[i][b]);
        printf("\t ");
        for(int b = 0; b < M; b++) printf("%2d ",max[i][b]);
        printf("\t  ");
        for(int b = 0; b < M; b++) printf("%2d ",need[i][b]);
        printf("\n");
    }
}
void bankers()
{
    dead = 0;
    for(int i = 0; i < M; i++) {
        tavail[i] = available[i];
    }
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < M; j++) {
            talloc[i][j] = allocated[i][j];
            tneed[i][j] = need[i][j];
        }
    }
    c = 0; no = 0;
    for(int i = 0; i < N; i++) overp[i] = 0;
    while(c < N) {
        no = 1;
        for(int i = 0; i < N; i++)
        {
            ncheck = 0;
            if(!overp[i]) {
                for(int j = 0; j < M; j++)
                {
                    diff[j] = tavail[j] - tneed[i][j];
                    if(diff[j] < 0) ncheck = 1;
                }
                if(ncheck != 1) {
                    for(int k = 0; k < M; k++) {
                        tavail[k] += talloc[i][k];
                    }
                    printf("P%d is over!\n", i+1);
                    overp[i] = 1;
                    safety[c] = i+1;
                    c++;
                    no = 0;
                }
            }
        }
        if(no == 1) {
            printf("\nDeadlock!\n");
            dead = 1;
            for(int k = 0; k < N; k++)
                if(overp[k] != 1) printf("P%d ",k+1);
            printf("\n");
            break;
        }
    }
    printf("\n");
    if(c == N)
    {
        printf("Safety: ");
        for(int i = 0; i < N; i++) printf("P%d ", safety[i]);
    }
    printf("\n");
}
void request()
{
    int p, r[10], can = 1;
    printf("Process: ");
    scanf("%d", &p);
    printf("Request for: ");
    for(int i = 0; i < M; i++) {
        scanf("%d", &r[i]);
    }
    for(int i = 0; i < M; i++) {
        if(available[i] - r[i] < 0) can = 0;
    }
    if(can == 0) printf("\nCannot request!\n");
    else {
        for(int i = 0; i < M; i++) {
            allocated[p-1][i] += r[i];
            need[p-1][i] -= r[i];
            available[i] -= r[i];
        }
        print();
        bankers();
        if(dead == 1) {
            for(int i = 0; i < M; i++) {
                allocated[p-1][i] -= r[i];
                need[p-1][i] += r[i];
                available[i] += r[i];
            }
        }
    }
}
void menu()
{
    printf("1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit\
nChoice: ");
    scanf("%d", &ch);
    if(ch == 1) read();
    else if(ch == 2) print();
    else if(ch == 3) bankers();
    else if(ch == 4) request();
    else exit(0);
}
int main()
{
    while(1) menu();
    return 0;
}
praveen@praveen gcc Bankers.c
praveen@praveen ./a.p##[Kout
```

```
1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 1
No. of resources and processes: 4 4
Total no. of resource 1: 11
Total no. of resource 2: 11
Total no. of resource 3: 8
Total no. of resource 4: 10
Process 1:
Allocated: 2 0 0 1
Maximum: 4 2 1 2
Process 2:
Allocated: 3 2 # ## #1 2 1
Maximum: 5 2 5 2
Process 3:
Allocated: 2 0 1 3
Maximum: 2 3 1 6
Process 4:
Allocated: 2 7 4 4
Maximum: 4 10 6 5
1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 2
Data:
P     Alloc        Max          Need               Available
      R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4
                                                     2  3  1  1
1      2  0  0  1         4  2  1  2         2  2  1  1
2      3  1  2  1         5  2  5  2         2  1  3  1
3      2  0  1  3         2  3  1  6         0  3  0  3
4      2  7  4  4         4 10  6  5         2  3  2  1
1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 3
P1 is over!

Deadlock!
P2 P3 P4

1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 1
No. of resources and processes: 4 5
Total no. of resource 1: 11
Total no. of resource 2: 11
Total no. of resource 3: 8
Total no. of resource 4: 10
Process 1:
Allocated: 2 0 0 1
Maximum: 4 2 1 2
Process 2:
Allocated: 3 1 2 1
Maximum: 5 2 5 2
Process 3:
Allocated: 2 1 0 3
Maximum: 2 3 1 6
Process 4:
Allocated: 1 3 1 2
Maximum: 1 4 2 4
Process 5:
Allocated: 1 4 3 2
Maximum: 3 6 6 5
1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 2
Data:
P     Alloc        Max          Need               Available
      R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4
```

```
                                                    2  2  2  1
1      2  0  0  1        4  2  1  2        2  2  1  1
2      3  1  2  1        5  2  5  2        2  1  3  1
3      2  1  0  3        2  3  1  6        0  2  1  3
4      1  3  1  2        1  4  2  4        0  1  1  2
5      1  4  3  2        3  6  6  5        2  2  3  3
1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 3
P1 is over!
P4 is over!
P5 is over!
P2 is over!
P3 is over!

Safety: P1 P4 P5 P2 P3
1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 4
Process: 5
Request for: 0 0 2 0
Data:
P     Alloc        Max          Need              Available
      R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4
                                                    2  2  0  1
1      2  0  0  1        4  2  1  2        2  2  1  1
2      3  1  2  1        5  2  5  2        2  1  3  1
3      2  1  0  3        2  3  1  6        0  2  1  3
4      1  3  1  2        1  4  2  4        0  1  1  2
5      1  4  5  2        3  6  6  5        2  2  1  3

Deadlock!
P1 P2 P3 P4 P5


1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 4
Process: 2
Request for: 1 1 0 0
Data:
P     Alloc        Max          Need              Available
      R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4        R1 R2 R3 R4
                                                    1  1  2  1
1      2  0  0  1        4  2  1  2        2  2  1  1
2      4  2  2  1        5  2  5  2        1  0  3  1
3      2  1  0  3        2  3  1  6        0  2  1  3
4      1  3  1  2        1  4  2  4        0  1  1  2
5      1  4  3  2        3  6  6  5        2  2  3  3

Deadlock!
P1 P2 P3 P4 P5


1. Enter data 2. Print data 3. View Sequence 4. Request 5. Exit
Choice: 5
praveen@praveen exit
exit

Script done on 2019-03-05 23:42:02+0530
```