|  |  |
| --- | --- |
| **Name:** | Bodhisatya Ghosh |
| **Class:** | CSE DS |
| **Batch:** | B |
| **UID:** | 2021700026 |
| **Experiment:** | 9 |

**Aim**: Write a program for preventing race conditions and deadlock avoidance for the banker’s algorithm as follows. OS has various resources. Customer will request the particular resource. The OS will release and grant the product only if it leaves the system in safe state. A request that leaves the federation in an unsafe state will be denied. Take the Allocation and Available from user. Print the Allocation, Max and Available. Find and print the Need. Find the safe sequence if any and print it and tell whether federation is in safe state or not. Take a request from user and tell whether this request will be granted immediately or not.  
  
**Code:**

#include <stdio.h>

#include <stdlib.h>

int alloc[5][3];

int maxneed[5][3];

int remneed[5][3];

int total[3];

int avail[3];

int processed[5]={0};

void initialize(int alloc[5][3], int total[3], int maxneed[5][3], int remneed[5][3])

{

    for(int k=0;k<3;k++) //total resources matrix

    {

        printf("Enter number of total RESOURCE %d instances: ",k+1);

        scanf("%d",&total[k]);

        avail[k]=total[k];

    }

    for(int i=0;i<5;i++)

    {

        printf("ALLOCATION for PROCESS %d:\n",i+1);

        for(int j=0;j<3;j++)

        {

            printf("Enter number of RESOURCE %d instances that are being used: ",j+1);

            scanf("%d",&alloc[i][j]);

        }

        printf("\n");

    }

    for(int i=0;i<5;i++) //max need matrix

    {

        printf("Maximum needed instances for PROCESS %d:\n",i+1);

        for(int j=0;j<3;j++)

        {

            printf("Enter MAX number of RESOURCE %d instances that will be needed: ",j+1);

            scanf("%d",&maxneed[i][j]);

        }

    }

    printf("Calculating number of REMAINING NEEDE resources\n");

    for(int i=0;i<5;i++) //calculating remaining need matrix

    {

        for(int j=0;j<3;j++)

        {

            remneed[i][j]=maxneed[i][j]-alloc[i][j];

        }

    }

}

void print\_1darray(int n,int arr[n])

{

    for(int i=0;i<n;i++)

    {

        printf("%d ",arr[i]);

    }

    printf("\n");

    printf("\n");

}

void print\_2darray(int m, int n, int arr[m][n])

{

    for(int i=0;i<m;i++)

    {

        printf("P%d: ",i+1);

        for(int j=0;j<n;j++)

        {

            printf("%d ",arr[i][j]);

        }

        printf("\n");

    }

    printf("\n");

}

void print(int alloc[5][3], int total[3], int maxneed[5][3], int remneed[5][3])

{

    printf("Total resources: \n");

    print\_1darray(3,total);

    printf("Allocated: \n");

    print\_2darray(5,3,alloc);

    printf("Max needed: \n");

    print\_2darray(5,3,maxneed);

    printf("Remaining needed: \n");

    print\_2darray(5,3,remneed);

}

int request(int num)

{

    int flag=1;

    for(int i=0;i<3;i++)

    {

        if(remneed[num][i]>avail[i])

        {

            return 0;

            //printf("Request cannot be granted");

            //break;

        }

    }

    return 1;

    /\*if(flag==1)

    {

        printf("Request can be granted");

    }\*/

}

void grant(int num)

{

    for(int i=0;i<3;i++)

    {

        avail[i]=avail[i]+alloc[num][i];

        alloc[num][i]=0;

        remneed[num][i]=0;

    }

}

void safesequence()

{

    int conf=0,granted=0;

    for(int a=0;a<5;a++)

    {

        if(request(a))

        {

            conf++;

        }

    }

    if(conf==0)

    {

        printf("No safe sequence");

    }

    else

    {

        printf("Safe sequence is: ");

        while(granted<5)

        {

            for(int i=0;i<5;i++)

            {

                if(processed[i])

                {

                    continue;

                }

                if(request(i))

                {

                    grant(i);

                    printf("P%d -->",i);

                    processed[i]=1;

                    granted++;

                }

            }

        }

    }

}

void main()

{

    int customer;

    initialize(alloc,total,maxneed,remneed);

    print(alloc,total,maxneed,remneed);

    for (int i = 0; i < 3; i++)

    {

        for (int j = 0; j < 5; j++)

        {

            avail[i]=avail[i]-alloc[j][i];

        }

    }

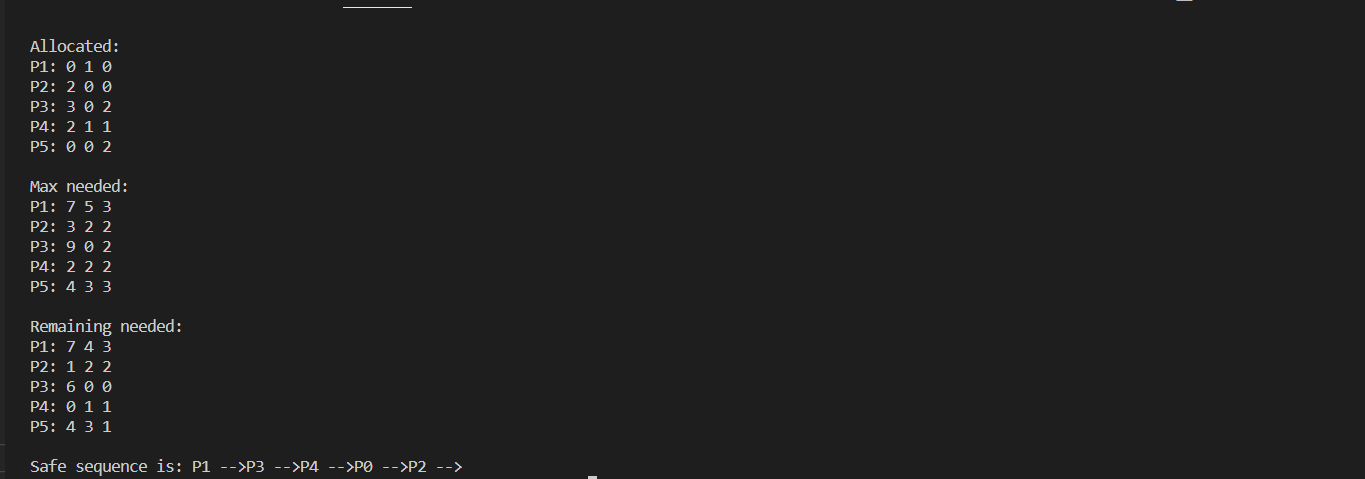
    safesequence();

    /\*printf("Enter PROCESS NUMBER which will request resources: ");

    scanf("%d",&customer);\*/

}

**Output:**

**  
  
Conclusion:** In this experiment we have learnt how to implement banker’s algorithm.