1. **Bankers Safety Search with Resource Request:**

Input:

#include<iostream>

using namespace std;

int n,m,i,j,p[10],alloc[10][10],Max[10][10],need[10][10],avail[10],work[10],finish[10],sequence[10],request[10],check1,check2=0,preq,check3,check4,Count=0,k=1;

void input()

 {

    cout<<"Enter the number of processes: ";

    cin>>n;

    cout<<"Enter the number of resources: ";

    cin>>m;

    for(i=1;i<=n;i++)

    {

        cout<<"Enter allocation for process["<<i<<"]: ";

        for(j=1;j<=m;j++)

        {

            cin>>alloc[i][j];

        }

    }

      cout<<endl;

      for(i=1;i<=n;i++)

      {

        cout<<"Enter max for process["<<i<<"]: ";

        for(j=1;j<=m;j++)

        {

            cin>>Max[i][j];

        }

      }

      cout<<endl<<"Enter the available vector: ";

      for(i=1;i<=m;i++)

      {

        cin>>avail[i];

      }

      cout<<endl;

      cout<<"The need matrix is: "<<endl;

      for(i=1;i<=n;i++)

      {

        for(j=1;j<=m;j++)

        {

            need[i][j]=Max[i][j]-alloc[i][j];

            cout<<need[i][j]<<" ";

        }

         cout<<endl;

      }

}

void safety\_seq()

{

    for(i=1; i<=m; i++)

    {

        work[i] = avail[i];

    }

    for(i=1; i<=n; i++)

    {

        finish[i] = 0;

    }

    while (Count<n)

    {

        for(i=1; i<=n; i++)

        {

            check1 = 0;

            for(j=1; j<=m; j++)

            {

                if((finish[i]==0) && (need[i][j]<=work[j]))

                {

                    check1++; //for number of resources

                }

                if(check1==m)

                {

                    //process is true,need to update work vector

                    for(j=1; j<=m; j++)

                    {

                        work[j] += alloc[i][j];

                        finish[i] = 1;

                    }

                    sequence[k] = i;

                    k++;

                    Count++;

                }

            }

        }

        for(i=1; i<=n; i++)

        {

            if(finish[i]==1)

            {

                check2++; //for number of processes

            }

            if(check2==n)

            {

                cout<<"\nSafe sequence exists!\n";

                cout<<"\nSafety sequence is:\n{ ";

                for(i=1; i<=n; i++)

                {

                    cout<<"P"<<sequence[i]<<" ";

                }

                cout<<"}";

            }

        }

    }

    if (check2 != n)

    {

        cout << "\nThere is a deadlock. Safety sequence does not exist!";

    }

  }

int resource\_request()

 {

    cout<<endl<<"\nEnter the process from which request is coming: ";

    cin>>preq;

    cout<<endl<<"Enter the request: ";

    for(j=1;j<=m;j++)

     {

        cin>>request[j];

     }

     for(j=1;j<=m;j++)

      {

        if(request[j]<=need[preq][j])

         {

            check3++;

         }

      }

      if(check3==m)

       {

         for(j=1;j<=m;j++)

          {

            if(request[j]<=avail[j])

             {

                check4++;

             }

          }

          if(check4==m)

           {

            cout<<endl<<"Request can be granted. Resources are available!";

            for(j=1;j<=m;j++)

             {

                alloc[preq][j]+=request[j];

                need[preq][j]-=request[j];

                avail[j]-=request[j];

             }

             cout<<endl<<"\nAllocation: ";

             for(j=1;j<=m;j++)

              {

                cout<<alloc[preq][j]<<" ";

              }

              cout<<"\nNeed: ";

              for(j=1;j<=m;j++)

               {

                cout<<need[preq][j]<<" ";

               }

               cout<<"\nAvailable: ";

              for(j=1;j<=m;j++)

               {

                cout<<avail[j]<<" ";

               }

           }

          else

           {

            cout<<endl<<"Request cannot be granted. Resources are not available!";

           }

       }

      else

       {

        cout<<endl<<"Request cannot be granted as it is exceeding its maximum claim.";

       }

       return 0;

 }

 int main()

  {

    input();

    safety\_seq();

    resource\_request();

    return 0;

  }

Output:

Enter the number of processes: 5

Enter the number of resources: 4

Enter allocation for process[1]: 0 0 1 2

Enter allocation for process[2]: 2 0 0 0

Enter allocation for process[3]: 0 0 3 4

Enter allocation for process[4]: 2 3 5 4

Enter allocation for process[5]: 0 3 3 2

Enter max for process[1]: 0 0 1 2

Enter max for process[2]: 2 7 5 0

Enter max for process[3]: 6 6 5 6

Enter max for process[4]: 4 3 5 6

Enter max for process[5]: 0 6 5 2

Enter the available vector: 2 1 0 0

The need matrix is:

0 0 0 0

0 7 5 0

6 6 2 2

2 0 0 2

0 3 2 0

Safe sequence exists!

Safety sequence is:

{ P1 P4 P5 P2 P3 }

Enter the process from which request is coming: 3

Enter the request: 0 1 0 0

Request can be granted. Resources are available!

Allocation: 0 1 3 4

Need: 6 5 2 2

Available: 2 0 0 0

1. **Deadlock Detection:**

Input:

//Deadlock Detection

#include <iostream>

using namespace std;

int n, m, i, j, p[10], alloc[10][10], req[10][10], avail[10], work[10], finish[10], sequence[10], check1, check2;

void input()

 {

    cout<<"Enter the number of processes: ";

    cin>>n;

    cout<<"Enter the number of resources: ";

    cin>>m;

     for(i=1;i<=n;i++)

      {

        cout<<"Enter allocation for process["<<i<<"]: ";

        for(j=1;j<=m;j++)

         {

            cin>>alloc[i][j];

         }

      }

      cout<<endl;

      for(i=1;i<=n;i++)

       {

          cout<<"Enter request for process["<<i<<"]: ";

          for(j=1;j<=m;j++)

           {

             cin>>req[i][j];

           }

       }

      cout<<endl<<"Enter the available vector: ";

      for(i=1;i<=m;i++)

       {

        cin>>avail[i];

       }

}

void display()

{

    cout<<"\n\nProcess\tAllocation Request\n";

    for(i=1;i<=n;i++)

    {

        cout<<"P"<<i<<"\t";

        for(j=1;j<=m;j++)

        {

            cout<<alloc[i][j]<<" ";

        }

        cout<<"\t   ";

        for(j=1;j<=m;j++)

        {

            cout<<req[i][j]<<" ";

        }

        cout<<endl;

    }

}

int safeSequence()

{

    int count=0, iteration=0;

    for(i=1;i<=n;i++)

    {

        finish[i] = 0;

    }

    for(j=1;j<=m;j++)

    {

        work[j]=avail[j];

    }

    while(iteration<n)

    {

        for(i=1;i<=n;i++)

        {

            if(!finish[i])

            {

                check1 = 1;

                for(j=1;j<=m;j++)

                {

                    if(req[i][j]>work[j])

                    {

                        check1=0;

                        break;

                    }

                }

                if(check1)

                {

                    finish[i]=1;

                    for(j=1;j<=m;j++)

                    {

                        work[j]+=alloc[i][j];

                    }

                    count++;

                    sequence[count] = i;

                }

            }

        }

        iteration++;

    }

    for(i=1;i<=n;i++)

    {

        if(finish[i]==0)

        {

            return 0;

        }

    }

    return 1;

}

int resReq()

{

    int preq;

    cout<<"\nEnter process number requesting resource: ";

    cin>>preq;

    cout<<"Enter new request vector: ";

    for(j=1;j<=m;j++)

    {

        cin>>req[preq][j];

    }

    return safeSequence();

}

int main()

{

    input();

    display();

    if(safeSequence())

    {

        cout<<"\nSafe sequence exists.";

        cout<<"\nSafe sequence: ";

        for(i=1;i<=n;i++)

        {

            cout<<"P"<<sequence[i]<<" ";

        }

    }

    else

    {

        cout<<"\nDEADLOCK: No safe sequence.";

    }

    int c = resReq();

    switch(c)

    {

        case 0: cout<<"\nSystem is in deadlock.";

                break;

        case 1: cout<<"\nSafe sequence: ";

        for(i=1;i<=n;i++)

        {

            cout<<"P"<<sequence[i]<<" ";

        }

        break;

        default: cout<<"\nInvalid input.";

    }

}

Output:

Enter the number of processes: 5

Enter the number of resources: 3

Enter allocation for process[1]: 0 1 0

Enter allocation for process[2]: 2 0 0

Enter allocation for process[3]: 3 0 3

Enter allocation for process[4]: 2 1 1

Enter allocation for process[5]: 0 0 2

Enter request for process[1]: 0 0 0

Enter request for process[2]: 2 0 2

Enter request for process[3]: 0 0 0

Enter request for process[4]: 1 0 0

Enter request for process[5]: 0 0 2

Enter the available vector: 0 0 0

Process Allocation Request

P1 0 1 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

Safe sequence exists.

Safe sequence: P1 P3 P4 P5 P2

Enter process number requesting resource: 3

Enter new request vector: 0 0 1

System is in deadlock.