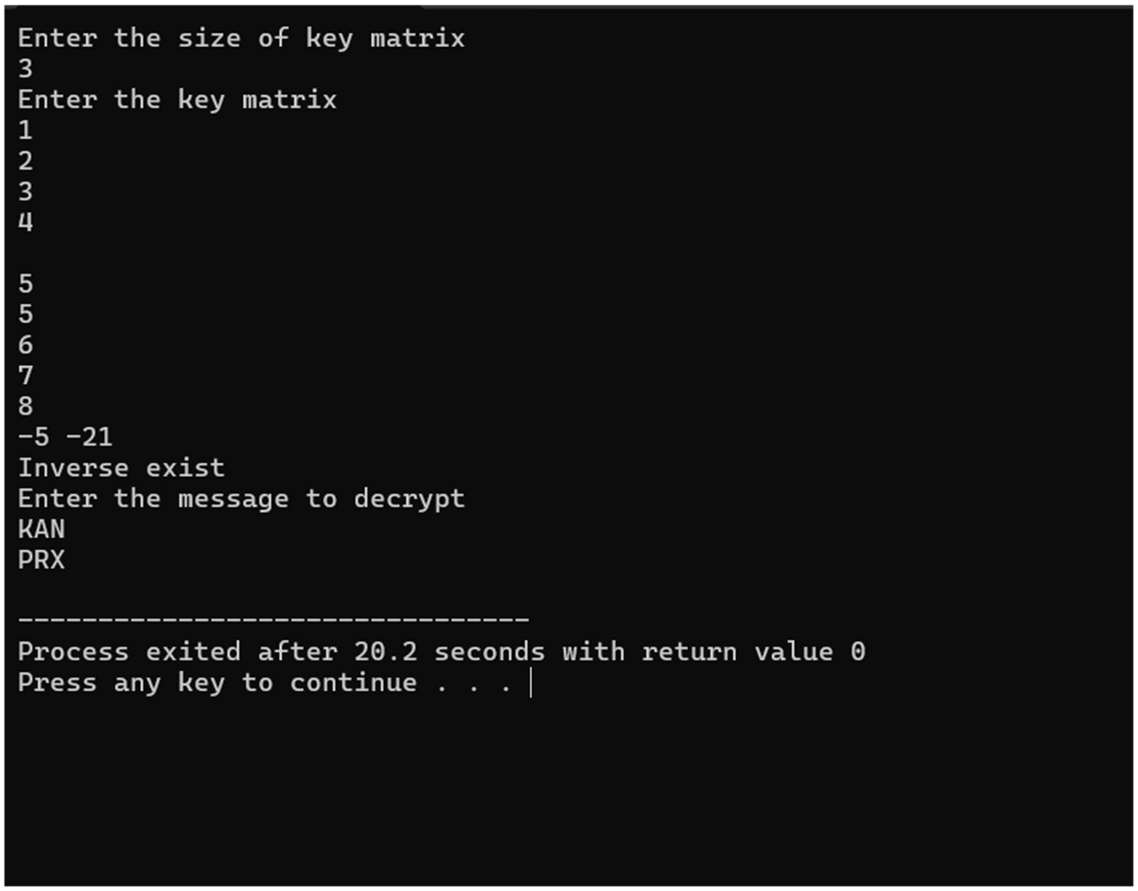
Information Security

Name: KANISH
College Roll No: CSC/21/53
University Roll No: 21059570017

6. Implement hill cipher substitution operation

```
#include<iostream>
#include<vector>
using namespace std;

int modInverse(int a, int m){
    a=a%m;
    for(int x=-m;x<m;x++)
       if((a*x)%m==1)
          return x;
}

void getCofactor(vector<vector<int> > &a, vector<vector<int> > &temp,
int p, int q, int n){
    int i=0,j=0;
    for(int row=0;row<n;row++){
        for(int col=0;col<n;col++){
            if(row!=p&&col!=q){
                temp[i][j++] = a[row][col];
                if (j==n-1){
                    j=0;
                    i++;
                }
            }
        }
    }
}

int determinant(vector<vector<int> > &a, int n, int N){
    int D = 0;
    if(n==1)
```

```cpp
        return a[0][0];
    vector<vector<int> > temp(N, vector<int>(N));
    int sign = 1;
    for(int f=0;f<n;f++){
        getCofactor(a, temp, 0, f, n);
        D += sign * a[0][f] * determinant(temp, n - 1, N);
        sign = -sign;
    }
    return D;
}

void adjoint(vector<vector<int> > &a,vector<vector<int> > &adj,int N){
    if(N == 1){
        adj[0][0] = 1;
        return;
    }
    int sign = 1;
    vector<vector<int> > temp(N, vector<int>(N));
    for(int i=0;i<N;i++){
        for(int j=0;j<N;j++){
            getCofactor(a, temp, i, j, N);
            sign = ((i+j)%2==0)? 1: -1;
            adj[j][i] = (sign)*(determinant(temp, N-1 , N));
        }
    }
}

bool inverse(vector<vector<int> > &a, vector<vector<int> > &inv, int
N){
    int det = determinant(a, N, N);
    if(det == 0){
        cout << "Inverse does not exist";
        return false;
    }
    int invDet = modInverse(det,26);
    cout<<det%26<<' '<<invDet<<'\n';
    vector<vector<int> > adj(N, vector<int>(N));
    adjoint(a, adj, N);
    for(int i=0;i<N;i++)
        for(int j=0;j<N;j++)
            inv[i][j] = (adj[i][j]*invDet)%26;
    return true;
}


int main(){
    int x,y,i,j,k,n;
    cout<<"Enter the size of key matrix\n";
    cin>>n;
    cout<<"Enter the key matrix\n";
    vector<vector<int> > a(n, vector<int>(n));
    vector<vector<int> > adj(n, vector<int>(n));
    vector<vector<int> > inv(n, vector<int>(n));

    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cin>>a[i][j];
        }
    }
    if(inverse(a,inv,n)){
        cout<<"Inverse exist\n";
    }

    cout<<"Enter the message to decrypt\n";
    string s;
    cin>>s;
    k=0;
    string ans;
    while(k<s.size()){
        for(i=0;i<n;i++){
            int sum = 0;
            int temp = k;
            for(j=0;j<n;j++){
                sum += ((inv[i][j] + 26)%26*(s[temp++]-'a')%26)%26;
                sum = sum%26;
            }
            ans+=(sum+'a');
        }
        k+=n;
    }
    //ans+='\0';
    int f=ans.size()-1;
    while(ans[f]=='x'){
        f--;
    }

    for(i=0;i<=f;i++){
        cout<<ans[i];
    }
    cout<<'\n';
    return 0;
}
```



```
Enter the size of key matrix
3
Enter the key matrix
1
2
3
4

5
5
6
7
8
-5 -21
Inverse exist
Enter the message to decrypt
KAN
PRX

--------------------------------
Process exited after 20.2 seconds with return value 0
Press any key to continue . . .
```