```
Define CharPoly(A)
 If Len(A)=Len(Transposed(A)) Then
  F:=Det(A-x*Identity(Len(A)));
  Return(F);
 Else
  Print("Keine quadratische Matrix!");
  Return(0);
 EndIf;
EndDefine;


Define EigenwertSystem(A)
 F:=CharPoly(A);
 G:=Factor(F);
 K:=[P In G | Deg(P[1])=1];
 N:=[P[1] | P In K];
 Y:=[];
 ForEach E In G Do
  Append(Y,E[2]);
 EndForEach;
 L:=[-CoeffOfTerm(1,O)/LC(O)|O In N];
 Z:=[];
 For I:=1 To Len(L) Do
  Append(Z,[L[I],Y[I]]);
 EndFor;
 Return(Z);
EndDefine;


Define EigenvectorSystem(A);
 L:=EigenwertSystem(A);
 M:=[];
 ForEach I In L Do
  Append(M,Flatten(LinKer(A-I[1]*Identity(Len(A)))));
 EndForEach;
 Return(M);
EndDefine;


Define EigenvectorSystemModP(A);
 B:=BringIn(A);
 L:=EigenwertSystem(B);
 M:=[];
 ForEach I In L Do
  Append(M,Flatten(LinKerModP(B-I[1]*Identity(Len(B)))));
 EndForEach;
 Return(M);
EndDefine;
```

```
Define MinPoly(A)
 If Len(A)<>Len(Transposed(A)) Then
  Print("Keine quadratische Matrix!");
  Return(0);
 EndIf;
 If IsZero(A) Then
  Return(x);
 EndIf;
 S:=-1;
 E:=[];
 MinPoly:=0;
 Repeat
  S:=S+1;
  D:=List(A^S);
  Append(E,Flatten(D));
  M:=Mat(E);
  MM:=Transposed(M);
  V:=LinKer(MM);
 Until V<>[];
 VV:=Flatten(V);
 For X:=1 To Len(VV) Do
  MinPoly:=MinPoly+(VV[X]*x^(X-1)/VV[Len(VV)]);
 EndFor;
 Return(MinPoly);
EndDefine;


Define ZSOperation(A,I,J)
 If IsSymmetric(A) Then
  E:=Identity(Len(A));
  If IsZero(A[I,I]) Then
   If IsZero(A[J,J]) Then
    X:=1;
    While X <= Len(A) And A[I,X]=0 Do
     X:=X+1;
    EndWhile;
    E[I,X]:=1;
    A:=E*A*Transposed(E);
   Else
    Tmp:=I;
    I:=J;
    J:=Tmp;
   EndIf;
  EndIf;
  E:=Identity(Len(A));
  E[J,I]:=-A[I,J]/A[I,I];
  D:=E*A*Transposed(E);
  Return(D);
 Else
  Print("Keine symmetrische Matrix!");
  Return(0);
 EndIf;
EndDefine;
```

```
Define ZSAlgorithmus(A);
 For N:=1 To (Len(A)-1) Do
  For I:=N+1 To Len(A) Do
   B:=ZSOperation(A,N,I);
   A:=B;
  EndFor;
 EndFor;
 Return(A);
EndDefine;


Define IsPosDefinite(A)
 If IsSymmetric(A) Then
  P:=TRUE;
  I:=1;
  While I <= Len(A) And P=TRUE Do
   If Det(Submat(A,1..I,1..I)) <= 0 Then
    P:=FALSE;
   EndIf;
   I:=I+1;
  EndWhile;
  Return(P);
 Else
  Print("Keine symmetrische Matrix!");
  Return(FALSE);
 EndIf;
EndDefine;


Define ExtEuklid(A,B)
 If A=0 And B=0 Then
  Return([0,0,0]);
 Elsif A=0 And B<>0 Then
  Return([0,Abs(B)/B,Abs(B)]);
 Elsif A<>0 And B=0 Then
  Return([Abs(A)/A,0,Abs(A)]);
 EndIf;
 V:=[Abs(A)/A,0,Abs(A)];
 W:=[0,Abs(B)/B,Abs(B)];
 If W[3]>V[3] Then
  X:=W;
  W:=V;
  V:=X;
 EndIf;
 R:=Mod(V[3],W[3]);
 While R<>0 Do
  Q:=1;
  While V[3]<>(Q*W[3]+R) Do
   Q:=Q+1;
  EndWhile;
  U:=[V[1]-Q*W[1],V[2]-Q*W[2],R];
  V:=W;
  W:=U;
  R:=Mod(V[3],W[3]);
 EndWhile;
 Return(W);
EndDefine;
```