```
> Berkowitz := proc(A::Matrix,x::name)
> local n,r,i,j,k,Ar,Cr,R,S,Q,C,Cn;
> uses LinearAlgebra;
>     n := RowDimension(A);
>     if ColumnDimension(A)<>n then error "matrix must be square" fi;
>     if n=1 then return A[1,1]-x; fi;
>     r := n-1;
>     Ar := A[1..r,1..r];
>     R := A[n,1..r];
>     S := A[1..r,n];
>     Cr := Berkowitz(Ar,x);
>     # compute Q[i] = R^T Ar^i S for 0 <= i <= r-1
>     Q[0] := expand(R.S);
>     for i to r-1 do S := Ar.S; Q[i] := R.S; od;
>     C := Array(0..r,[seq( coeff(Cr,x,i), i=0..r )]);
>     Cn := Cr*(A[n,n]-x) + add(add(C[k+j]*Q[j]*x^(k-1),j=0..r-k),k=1..r);
>     expand(Cn);
> end:

> with(LinearAlgebra):
> A := Matrix([[1,2,3],[5,7,8],[9,8,5]]);
                                    [1    2    3]
                                    [           ]
                               A := [5    7    8]
                                    [           ]
                                    [9    8    5]

> Berkowitz(A,x);
                               3       2
                             -x  + 13 x  + 54 x - 4

> CharacteristicPolynomial(A,x); # det( x I - A )
                              3       2
                             x  - 13 x  - 54 x + 4

> r := n-1;
                                   r := n - 1

> s := expand(sum(sum(1,j=0..r-k),k=1..r));
                                        2
                              s := 1/2 n  - 1/2 n

> re := M(n) = M(n-1)+(r-1)*(r^2+r+1)+s;
                                                                   2
                                                         2        n
        re := M(n) = M(n - 1) + (n - 2) ((n - 1)  + n) + ---- - n/2
                                                                  2

> expand(rsolve({re,M(1)=0},M(n)));
                           4        3        2
                      1/4 n  - 1/3 n  + 1/4 n  - 7/6 n + 1
```