Programme 1 [For Theorem 5].

```
import numpy as np
import sympy
from scipy import integrate
import math

    def V(i,M):
    n=M-6*i
    V=np.zeros((6,n))
    for t in range(0,n):
        if t ==0:
            V[0,t] = 1
        else:
            V[0,t]=0
        if t ==0:
            V[1,t]=0
        else:
            V[1,t]=(t+1)*eg_J(t)
        if t==0:
            V[2,t]=3
        elif t==1:
            V[2,t]=4
        else:
            V[2,t]=0
            V[3,t]=t+1
            V[4,t]=(t+1)*((t+1)**2-9*(t+1)/5+61/25)
            V[5,t]=(t+1)*((t+1)**2-3*(t+1)+91/25)
        return V

    def W(i,j,M):
    n=M-6*i
    W=np.triu(np.ones(n))
    if j==i:
        for s in range(0,n):
            for t in range(s,n):
                W[s,t]=(t+2-s)*(t+1-s)/2
    else:
        for s in range(0,n):
            for t in range(s,n):
                W[s,t]=(6*j+t+1)*(t+2-s)*(t+1-s)/2
    return W

    def f(x,i):
    return x**(i+1)*math.sqrt(x/3-3/4*x**2+3/5*x**3-1/6*x**4)

    def g(x,i):
    return x**(i-1)/math.sqrt(x/3-3/4*x**2+3/5*x**3-1/6*x**4)

    def eg_I(i):
    x0 = (28+10*math.sqrt(10))**(1/3)/10-3/(5*(28+10*math.sqrt(10))
**(1/3))+6/5
    n,w=integrate.quad(f,0,x0,args=(i))
    return n

    def eg_J(i):
    x0 = (28+10*math.sqrt(10))**(1/3)/10-3/(5*(28+10*math.sqrt(10))
**(1/3))+6/5
    n,w=integrate.quad(g,0,x0,args=(i))
    return n

    def main(m):
    if (m-1)%6==0:
        k=int((m-1)/6)-1
    else:
        k=int((m-1)/6)
    M = m+1
    r=(m-1)%6
    P0=np.zeros((7,M))
    for j in range(0,M):
        P0[0,j]=eg_I(j)
        if j==0:
            P0[1,0]=1
        else:
            P0[1,j]=0
        if j==0:
            P0[2,0]=0
        else:
            P0[2,j] =eg_J(j)
        if j==0:
            P0[3,0]=3
        elif j==2:
            P0[3,1]=2
        else:
            P0[3,j] = 0
            P0[4,j]=1
            P0[5,j]=j**2+j/5+41/25
            P0[6,j]=(j-1)*j+41/25
    p = [1 for x in range(0,k+1)]
    for i in range (1,k+1):
        p[i]=V(i,M)
        for j in range (1,i+1):
            p[i]=np.dot(p[i],W(i,j,M))
    Q = [1 for x in range(0,k+1)]
    for i in range(1,k+1):
        Q[i]=np.append(np.zeros((6,6*i)),p[i],axis = 1)
    if k==0:
        H=P0
    else:
        for i in range(1,k+1):
            if i==1:
                H=np.append(P0,Q[i],axis = 0)
            else:
                H=np.append(H,Q[i],axis =0)
    if r==0:
        rk1 = np.linalg.matrix_rank(np.mat(H[0:(6*k+5),0: (m+1)]))
        rk2 = np.linalg.matrix_rank(np.mat(np.row_stack((H[0:(6*k+
3),0:(m+1)],H[6*k+4:6*k+6,0:(m+1)]))))
        print(6*k+5,rk1,rk2)
    elif r<4:
        rk1 = np.linalg.matrix_rank(np.mat(H[0:(6*k+r+1),0:(m+1)]))
        rk2 = np.linalg.matrix_rank(np.mat(np.row_stack((H[0:(6*k+
r),0:(m+1)],H[6*k+5,0:(m+1)]))))
        print(6*k+1+r,rk1,rk2)
    else:
        rk1 = np.linalg.matrix_rank(np.mat(H[0:(6*k+5),0: (m+1)]))
        rk2 = np.linalg.matrix_rank(np.mat(np.row_stack((H[0:(6*k+
3),0:(m+1)],H[6*k+4:6*k+6,0:(m+1)]))))
        print(6*k+5,rk1,rk2)


    if __name__ == "__main__":
    print("restart")
    for m in range(4,301):
        main(m)
```

Programme 2 [For Theorem 6].

```
import sympy
import numpy as np
from math import factorial as fac

    def binomial(x, y):
    try:
        binom = fac(x) \\ fac(y) \\ fac(x - y)
    except ValueError:
        binom = 0
```

```
return binom

def crearMatrix(name,N):
X = [ ]
for i in range(N+1):
    row = [ ]
    for j in range(N+1):
        row.append(sympy.Symbol("c"+'_'+str(i)+'_'+ str(j)))
    X.append(row)
return sympy.Matrix(X)

def Myvertor(m,N):
c = crearMatrix("c",N)
c_new = c
    for i in range(0,N+1):
    for j in range(0,N+1):
        s= i+j;
        if s>=m:
            c_new[i,j] = 0
c_end = [ ]
for s in range(0,m):
for i in range(0,s+1):
    j = s-i;
    if j%2==0:
        a =c_new[i,j]
        c_end.append(a)
return c_end,c_new

def dfac(m):
if m <= 0:
    y= 1
else:
    y=m *dfac(m-2)
return y

def dm(n):
l = int((n+1)/2);
d =0;
for j in range(l,n+1):
    d = d+binomial(j,n-j)
return d

def A2(m,k):
N =m-3
    if k==1:
    A=np.zeros([3,N])
    for i in range(0,3):
        for j in range(0,N):
            if j>i+1:
                A[i,j]=0
            else:
                A[i,j]=(i+1)*dm(i+1-j)
else:
    A=np.zeros([4*k-4,N])
    for i in range(0,4*k-4):
        for j in range(0,N):
            if j>i+4:
                A[i,j]=0
            else:
                A[i,j]=(i+4)*dm(i+4-j)
return A

def A1(k):
if k==0:
    A=np.identity(3)
elif k==1:
    A=np.zeros([3,4])
    for i in range(0,3):
        for j in range(0,4):
            if j>i+1:
                A[i,j]=0
            else:
                A[i,j]=(i+1)*dm(i+1-j)
else:
    A= np.zeros([4*k-4,4*k])
    for i in range(0,4*k-4):
        for j in range(0,4*k):
        if j>i+4:
            A[i,j]=0
        else:
            A[i,j]=(i+4)*dm(i+4-j)
return A

def matrixU(i):
l1=1
l2=1
for j in range(1,i+1):
    l1 = l1*(4*j+3)
    l2 = l2*(4*j+5)
U = sympy.diag(4**i/l1,4**i/l2,1)
return U

def rk(m):
k = int((2*int(m/2)-2)/3)+int((m+1)/2)-1)
N = 4*k
C,d = Myvertor(m,N)
R = sympy.Matrix([[1,0,0],[46/25,-3/5,2],[46/25,0,2]])
F_end = sympy.Matrix([[1],[1]])
for l in range(0,k+1):
    if l==0:
        c_0 = sympy.Matrix(R)*sympy.Matrix([d[0,0],d[1,0],d[2,
0]]);
        F_end = np.concatenate((F_end,c_0),axis = 0)
    else:
        A_1 = A1(0)
        for j in range(0,l):
            A_1 = sympy.Matrix(A_1)*sympy.Matrix (A1(j))
        if l<=int((m-3)/4):
            A_1 = sympy.Matrix(A_1)*sympy.Matrix (A1(l))
            c_1 = [ ]
            for i in range(3,4*l+3):
                c_1.append(d[i,0])
            F_1 = sympy.Matrix(A_1)*sympy.Matrix(c_1)
        else:
            if m==3:
            F_1=np.zeros([3,1])
            else:
                A_1= sympy.Matrix(A_1)* sympy.Matrix(A2(m,l))
                c_2 = [ ];
                for i in range(3,m):
                    c_2.append(d[i,0])
                F_1 = sympy.Matrix(A_1)*sympy.Matrix (c_2)
        if l==1:
            c_3 = sympy.Matrix([d[0,2],d[1,2],d[2,2]])
            F_2=sympy.Matrix(A1(0))*sympy.Matrix (c_3)
        else:
            c_4 = sympy.Matrix([d[0,2*l],d[1,2*l],d[2,2*l]]);
            F_2 = dfac(2*l-1)*sympy.Matrix(c_4)
            for i in range(1,l):
                y= dfac(2*i-1)
                A_2 = A1(0)
                for j in range(1,l-i+1):
                    A_2 = sympy.Matrix(A_2)*sympy.Matrix(A1(j));
                c_5 = [ ];
                for h in range(3,(4*(l-i)+3)):
                    c_5.append(d[h,2*i])
```

```
            F_2 = F_2+y∗sympy.Matrix(A_2)∗sympy.Matrix(c_5)
        f = F_1+F_2
        U = matrixU(l)
        F = U∗R∗f
        F_end = np.concatenate((F_end,F),axis = 0)
F_end_1 = F_end[2:]
s = 3∗int((m+1)/2)+2∗int(m/2)-4
F_end_2= sympy.Matrix(F_end_1[0:s])
Jaco = F_end_2.jacobian(sympy.Matrix(C))
r=Jaco.rank( )
print("Rank is:",r)
return r

if __name__ == "__main__":
print("start")
p = 23
com_vertor = [ ]
for m in range(3,p):
    r = rk(m)
```