```
using Distributions
using QuadGK

#Erstellen Arrays für die Stäbe

#Wanddicke
t = [6.3 6.3 6.3 6.3 6.3 8.0 6.3 6.3 6.3 10.0]
#Querschnittsfläche
A = [3.77 3.77 3.77 3.77 3.77 4.70 3.77 3.77 3.77 5.74] * 10^-3
#Flächenträgheitsmoment
I = [1.46 1.46 1.46 1.46 1.46 1.78 1.46 1.46 1.46 2.10] * 10^-5
#Länge
L = [5.6 4.2 4.2 7.0 4.2 sqrt(35.28) 5.6 4.2 7.0 4.2]

#Stabkräfte (V=1)
N = [0 1 0 1.25 -0.75 -1.414 -1 0.75 0 -1.175]
E = 2.1e8

#Belastung
muh_x1 = 410
sigma_x1 = 70

#Material
muh_x2 = 30.2e4
sigma_x2 = 2.44e4
x02 = 19.9e4

#leere Arrays
cj = zeros(10)
Pfa = zeros(10)
Pfb = zeros(10)
Pfc = zeros(10)
PfcII = zeros(10)
cjk = zeros(10)
kj = zeros(10)
Pfa_ges = 0
Pfb_ges  = 0
Pfc_ges = 0

##################################################


#Koeffizienten cj und cjk/kj (Aufgabe c)
for i = 1:10
    cj[i] = -abs(N[i]/A[i])
    if N[i] < 0
        cjk[i] = -abs(N[i])
        kj[i] = E*I[i]*pi^2/L[i]^2
    end
end


#Werte a und b
a = 1/sigma_x1*pi/sqrt(6)
b = muh_x1 - 0.5772/a

#Berechnung von sigma_u und muh_u
sigma_u = sqrt(log(1+(sigma_x2/(muh_x2-x02))^2))
muh_u = log(sqrt((muh_x2-x02)/(1+sigma_x2/(muh_x2-x02))^2))

##################################################
```

```
# Funktionen f_x1 und F_min und fk
#Aufgabe a) und b)
function f_x1(x)
    return f_x1 = a*exp(-a*(x-b)*exp(-a*(x-b)))
end

function F_min_x1(x)
    z = (log(x-x02)-muh_u)/sigma_u #NN verteilt
    return pdf(Normal(), z)
end

#Aufgabe c) Fall I
function fk(k)
   z = (-1/sigma_x1*(-k-muh_x1)) #NV verteilt
   return pdf(Normal(),z)
end

#Aufgabe c) Fall II
function fkII(k)
   return exp(-exp(0.0321*k+10.6))
end

##################################################


#Aufgabe a)


for i = 1:10
    if cj[i] == -0.0
        Pfa[i] = 0
    else
        Pfa[i], error = quadgk(x -> F_min_x1(-cj[i]*x)*f_x1(x),-x02/cj[i],1234)
        global Pfa_ges = Pfa_ges + Pfa[i]
    end
end



#Aufgabe b)

for i = 1:10
   if cj[i] == -0.0

   else
       Pfb[i], error = quadgk(x -> f_x1(x) * F_min_x1(-cj[i]*x), -x02/cj[i] ,
10000)
       global Pfb_ges = Pfb_ges + Pfb[i]
   end
end

#Aufgabe c)

for i = 1:10
   if cjk[i] == 0
       Pfc[i] = 0.0
   else
       k = kj[i]/cjk[i]
       Pfc[i] = fk(k)
       PfcII[i] = 1 - fkII(k)
       global Pfc_ges = Pfc_ges + Pfc[i]
   end
end
```

```
##################################################
#Ausgabe
println("cj: ",cj)
println("cjk: ",cjk)
println("kj: ", kj)
println()
println("a: ",a,"  b: ", b)
println("sigma_u :",sigma_u,"   muh_u: ",muh_u)
println()
println("Aufgabe a)")
println("Pfa: ", Pfa)
println("Pfa_ges: ",Pfa_ges)
println()
println("Aufgabe b)")
println("Pfb: ", Pfb)
println("Pfb_ges: ", Pfb_ges)
println()
println("Aufgabe c)")
println("Fall I")
println("Pfc :", Pfc)
println("Pfc_ges: ", Pfc_ges)
println("Fall II")
println("PfcII :",PfcII)
```