```
import math

# 1030 HR Steel info
sultmet = 470_000_000
sulteng = 68
sy = 37_500

#finding Se prime using ultimate strength
seprime_met = sultmet/2
seprime_eng = sulteng/2

#finding sqrt a to use in converting kt and kts to kf and kfs

root_a_bending = 0.246-(3.08*10**(-3)*sulteng)+(1.51*10**(-5)*sulteng**2)-
(2.67*10**(-8)*sulteng**3)
root_a_torsion = 0.190-(2.51*10**(-3)*sulteng)+(1.35*10**(-5)*sulteng**2)-
(2.67*10**(-8)*sulteng**3)

#function to find kf and kfs
def stress_concentration(diameter,kt_criteria):

    if kt_criteria == 'sharp':
        kt = 2.7
        kts = 2.2
        r = .02 *diameter
    elif kt_criteria =='wide':
        kt = 1.7
        kts = 1.5
        r = .1 *diameter
    elif kt_criteria =='key':
        kt = 2.14
        kts = 3
        r =.02*diameter
    elif kt_criteria =='sled':
        kt = 1.7
        kts = 1
        r =.01
    elif kt_criteria == 'ring':
        kt = 5
        kts = 3
        r = .01

    kf = 1 + ((kt - 1)/(1 + (root_a_bending/math.sqrt(r))))
    kfs = 1 + ((kts - 1)/(1 + (root_a_torsion/math.sqrt(r))))
```

```
    return kf,kfs

#function to find endurance strength
def se(diameter,sult,seprime):

    ka = 2*(sult**(-.217))

    if diameter <= 2 and diameter != 0:
        kb = 0.879 * (diameter**(-0.107))
    elif diameter > 2:
        kb = 0.91*(diameter**(-.157))
    else:
        kb = 1

    se = seprime * ka * kb

    return se

#function to find sigma A prime and sigma M prime
def sigma_prime(diameter,ma,tm,kt_criteria):

    sigmaa_prime =
math.sqrt((((32*(stress_concentration(diameter,kt_criteria)[0])*ma)/(math.pi*(dia
meter**3)))**2))
    sigmam_prime =
math.sqrt(3*(((16*(stress_concentration(diameter,kt_criteria)[1])*tm)/(math.pi*(d
iameter**3)))**2))
    return (sigmaa_prime,sigmam_prime)

#function to solve safety factor using goodman
def goodman(d, kf, kfs, ma, tm, sult,seprime):

    goodman_se = se(d,sult,seprime)
    # finding A
    a = math.sqrt((4*((kf*ma)**2)))
    #finding B
    b = math.sqrt((3*((kfs*tm)**2)))
    safety = ((math.pi*(d**3))/16) * (((a/(goodman_se*1000))+(b/(sult*1000)))**(-
1))
    return safety

#function to solve safety factor using von_mises
def von_mises(kf,kfs,sy,ma,tm,d):
```

```
    sigma_max =
math.sqrt((((32*kf*ma)/(math.pi*(d**3)))**2)+(3*(((16*kfs*tm)/(math.pi*(d**3)))**
2)))
    safety = (sy/sigma_max)
    return safety

#function to solve safetyfactor using conservative approximation
def conservative(sigmaM,sigmaA,sy):

    safety = sy/(sigmaM + sigmaA)
    return safety

#function to solve for diameter checking goodman and conservative approximation
def cig(sy,ma,tm,sult,kt_criteria,seprime):

    good_diameter = 0
    conserve_diameter = 0
    good_safe = 0
    conserve_safe = 0

    #find diameter for goodman criteria
    while good_safe < 1.5:
        good_diameter += 0.001
        kf = stress_concentration(good_diameter,kt_criteria)[0]
        kfs = stress_concentration(good_diameter,kt_criteria)[1]

        good_safe = goodman(good_diameter,kf,kfs,ma,tm,sult,seprime)

    #find diameter for conservative approx
    while conserve_safe < 1.5:
        conserve_diameter += 0.001
        sigmaA = sigma_prime(conserve_diameter,ma,tm,kt_criteria)[0]
        sigmaM = sigma_prime(conserve_diameter,ma,tm,kt_criteria)[1]
        conserve_safe =conservative(sigmaA,sigmaM,sy)

    #comparing found diameters and returning biggest one found
    if conserve_diameter > good_diameter:
        return conserve_diameter
    else:
        return good_diameter

#variable to keep while loop running
run = True

#while loop for solving problems without having to start program over
```

```
while run == True:

    #inputs from problem
    bend_mom = int(input('what is the bending moment'))
    torque = int(input('what is the torque'))

    criteria = input('what criteria? (g, vm, c, cig)')
    kt_criteria= input('what is stress concentration?')


    if criteria != 'cig':
        diameter = float(input('what is the diameter'))

    if criteria =='g':
        print(goodman(diameter,stress_concentration(diameter,kt_criteria)[0],stre
ss_concentration(diameter,kt_criteria)[1],bend_mom,torque,sulteng,seprime_eng))
    elif criteria =='vm':
        print(von_mises(stress_concentration(diameter,kt_criteria)[0],stress_conc
entration(diameter,kt_criteria)[1],sy,bend_mom,torque,diameter))
    elif criteria =='c':
        print(conservative(sigma_prime(diameter,bend_mom,torque,kt_criteria)[1],s
igma_prime(diameter,bend_mom,torque,kt_criteria)[0],sy))

    elif criteria == 'cig':
        print(cig(sy,bend_mom,torque,sulteng,kt_criteria,seprime_eng))

    #getting out of while loop or continuing
    runagain = input('would you like to solve another? (y/n)')

    if runagain =='n':
        run = False
    else:
        run = True
```