```
#Merk at assert-statements ikke kreves i besvarelser, og kun
#er med for aa vise hvordan man kan sjekke at losning er korrekt
#3A
def vinnerlag(hjemmelag, bortelag, hjemmemaal, bortemaal):
    if hjemmemaal>bortemaal:
        return hjemmelag
    elif bortemaal>hjemmemaal:
        return bortelag
    else:
        return "uavgjort"

assert vinnerlag("Brann", "Molde", 2,3) == "Molde"
assert vinnerlag("Brann", "Molde", 1,0) == "Brann"
assert vinnerlag("Brann", "Molde", 2,2) == "uavgjort"

#3b
def forkort_lagliste(full):
    forkortet = []
    for lag in full:
        if not lag in forkortet:
            forkortet.append(lag)
    return forkortet


def forkort_lagliste_v2(full):
    return list(set(full))

assert forkort_lagliste(["Brann", "Molde", "Brann"]) == ["Brann", "Molde"]

#3C
def legg_inn_null_maal(lagliste):
    maal_liste = {}
    for lag in lagliste:
        maal_liste[lag] = 0
    return maal_liste

assert legg_inn_null_maal(["Brann", "Molde"]) == {"Brann" : 0, "Molde" : 0}

#3D
def ekstraher_lagliste(fn):
    lagliste = []
    for linje in open(fn):
        biter = linje.split()
        hjemmelag = biter[0]
        bortelag = biter[1]
        lagliste.append(hjemmelag)
        lagliste.append(bortelag)
    return lagliste
```

```
assert set(ekstraher_lagliste("IN1000_2018_fil3d.txt")) == set(["brann", "molde", "sarpsborg"])

#3e
def regn_poengsum(fn):
    lagliste = ekstraher_lagliste(fn)
    lagoversikt = legg_inn_null_maal(lagliste)

    for linje in open(fn):
        biter = linje.split()
        hjemmelag = biter[0]
        bortelag = biter[1]
        hjemmemaal = int(biter[2])
        bortemaal = int(biter[3])
        vinneren = vinnerlag(hjemmelag, bortelag, hjemmemaal, bortemaal)
        if vinneren==hjemmelag:
            lagoversikt[hjemmelag] += 3
        elif vinneren==bortelag:
            lagoversikt[bortelag] += 3
        else: #uavgjort
            lagoversikt[hjemmelag] += 1
            lagoversikt[bortelag] += 1
    return lagoversikt

assert regn_poengsum("IN1000_2018_fil3d.txt") == {"brann": 3, "molde":1, "sarpsborg":1}

#3f
def gull(lagoversikt):
    beste_lag = None
    maks_poeng = 0
    for lag in lagoversikt:
        poeng = lagoversikt[lag]
        if poeng>maks_poeng:
            beste_lag = lag
            maks_poeng = poeng
    return beste_lag

assert gull({"Brann" : 2, "Molde" : 3}) == "Molde"
assert gull({"Brann" : 4, "Molde" : 3}) == "Brann"

#3g
def finn_gull(fn):
    lagoversikt = regn_poengsum(fn)
    vinnerlag = gull(lagoversikt)
    print(vinnerlag)
    assert vinnerlag == "brann"

#5a
def godkjenn(familier):
```

```
    for familie in familier:
        finnes_myndig = False
        for medlem in familie:
            if medlem >= 18:
                finnes_myndig = True
        if not finnes_myndig:
            return False
    return True

def godkjenn_v2(familier):
    for familie in familier:
        if max(familie)<18:
            return False
    return True

assert godkjenn([  [10,2,30], [20,1] ])
assert not godkjenn([ [10,2,30], [10,1] ])
```