```python
def sjekk_reise(reise):
    forrige = reise[0][0] #komme seg fra ytre liste til indre lister
    for sted in reise: #sted er blokk, reise[0][1] -> reise[0][2]
        if not sted[0] == forrige:
            return False
        else:
            forrige = sted[1]

    return True


print(sjekk_reise([["Spania", "Frankrike"], ["Frankrike", "Norge"], ["Sverige", "Manila"]]))
```

```python
def fyll_til_ti(tallene):
    liste = tallene.copy()
    while len(liste) < 10:
        liste.append(0)

    return liste

print(fyll_til_ti([1, 2, 3, 4, 5]))
```

```python
def forkort_setning(streng):

    setning = "en krabbe skal en dag ut av skallet"
    ny_setning = ""
    for ord in setning.split(" "):
        if ord != streng:
            ny_setning += ord + " "

    return ny_setning

print(forkort_setning("en"))
print(forkort_setning("skal"))
```

```python
def finn_nest_stoerste(mengde):

    max_list = []
    for tall in mengde:
        max_list.append(tall)

    max_list.sort()

    return max_list[-2]

assert finn_nest_stoerste({11,5,34,19,0,0,18}) == 19
```

```python
def sjekk_om_fyord(setning, fyord, synonym_liste):

    biter = setning.split(" ")
    for bit in biter:
        for synonymer in synonym_liste:
            if bit in synonymer and fyord in synonymer:
                return True
    return False

assert sjekk_om_fyord("spis masse godsaker", "lemonade", [["saft","lemonade"],[
assert sjekk_om_fyord("spis masse godsaker", "snop", [["saft","lemonade"],["sno
assert sjekk_om_fyord("spis masse godsaker", "godsaker", [["saft","lemonade"],[
assert sjekk_om_fyord("spis masse godsaker", "godsaker", ["masse"]) == False
```

```python
def arverekke(ff, ek, ob):
    liste = []
    while ff in ob and ff != ek:
        liste.append(ff)
        ff = ob[ff]

    liste.append(ff)
    if not ek in liste:
        return []
    return liste


barn = {"Halfdan":"Harald", "Christian":"Hans", "Harald":"Eirik"}
personer = arverekke("Halfdan", "Eirik", barn)
print(personer)
```

```python
def arverekke(forfader, etterkommer, forstefodte):

    liste = []
    liste.append(forfader) #starter med forfader i lista
    for gjeldene in forstefodte:
        if gjeldene == liste[-1]: #først finne etterkommeren til halfdan
            liste.append(forstefodte[gjeldene]) #hvis gjeldene etterkommer også er en forfader,
            if forstefodte[gjeldene] == etterkommer: #bare interessert i at etterkommer er ekte
                return liste

    if liste[-1] == etterkommer: #hvis vi har funnet etterkommer
        return liste
    else:
        return [] #hvis ikke


barn = {"Halfdan":"Harald", "Christian":"Hans", "Harald":"Eirik"}
personer = arverekke("Halfdan", "Eirik", barn)
print(personer)
```

```python
class Hytte:

    def __init__(self, hyttenavn, ant_senger, pris):
        self._hyttenavn = hyttenavn
        self._ant_senger = ant_senger
        self._overnatting_pris = pris

    def hentNavn(self):
        return self._hyttenavn

    def totPris(self, antPersoner):
        return antPersoner * self._overnatting_pris

    def sjekkPlass(self, antPersoner):
        return self._ant_senger >= antPersoner

    def __str__(self):
        str = f"Hytte '{self._hyttenavn}' har {self._ant_senger} senger. "
        str += f"Det koster {self._overnatting_pris} per natt"
        return str

    def __eq__(self, annen):
        return self._hyttenavn == annen._hyttenavn #eller  == annen.hentNavn()
```

```python
class Tur:

    def __init__(self, hytteliste, tekst):
        self._hytter = hytteliste
        self._beskrivelse = tekst

    def skrivTur(self):
        print("\n" + self._beskrivelse)
        for hytte in self._hytter:
            print(hytte) #str metode

    def sjekkPris(self, antPersoner, maksBelop):

        totBelop = 0
        for hytte in self._hytter:
            if not hytte.sjekkPlass(antPersoner):
                return False
            totBelop += hytte.totPris(antPersoner)
            if totBelop > maksBelop:
                return False

        return True

        """
        for hytte in self._hytter:
            if hytte.sjekkPlass(antPersoner):
                if hytte.totPris(antPersoner) > maksBelop:
                    return True
        return False
        """

    def hentAntHytter(self):
        return len(self._hytter)
```

```python
class Turplanlegger:

    def __init__(self, hyttefil, turfil):
        self._hytter = self._hytterFraFil(hyttefil)
        self._turer = self._turerFraFil(turfil)

    def _hytterFraFil(self, filnavn):
        fil = open(filnavn, "r")
        hytter = {}
        for linje in fil:
            data = linje.strip().split()
            hyttenavn = str(data[0])
            antSenger = int(data[1])
            pris = float(data[2])
            hytter[hyttenavn] = Hytte(hyttenavn, antSenger, pris)

        fil.close()
        return hytter

    def _turerFraFil(self, filnavn):

        teller = 0
        fil = open(filnavn, "r")
        turer = []

        for linje in fil:
            if teller == 0:
                beskrivelse = linje.strip()
                teller += 1
            elif teller == 1:
                hData = linje.strip().split(" ")
                hytter_til_turen = []

                for hytte in hData:
                    if hytte in self._hytter:
                        hytter_til_turen.append(self._hytter[hytte])
                        #self._hytter[hytte] finner verdien som er referanse til
                        #Hytte-objekt i en ordbok
                turer.append(Tur(hytter_til_turen, beskrivelse))

                teller = 0

        return turer

    def finnTurer(self, antPersoner, maksBelop, maksDager):
        for tur in self._turer:
            if tur.hentAntHytter() <= maksDager:
                if tur.sjekkPris(antPersoner, maksBelop):
                    tur.skrivTur()

def main():

    turplaner = Turplanlegger("hytter.txt", "turer.txt")
    turplaner.finnTurer(7, 7000, 5)

main()
```

```python
from random import randint
class Gaard:

    def __init__(self):
        self._aaker = []
        self._generasjon = 0

        melon = Vannmelon()
        self._aaker.append(melon)

    def plant_vannmelon(self):
        nye_meloner = []
        for melon in self._aaker:
            for fro in range(melon.hent_ant_fro()): #range = lik mengde gjennomkjøringer som melonen har frø
                if randint(0,2) in [0,1]: #66%
                    ny_melon = Vannmelon()
                    nye_meloner.append(ny_melon)

        for melon in self._aaker:
            self._aaker.remove(melon)

        for melon in nye_meloner:
            self._aaker.append(melon)

        self.neste_generasjon()

    def neste_generasjon(self):
        self._generasjon += 1

    def __str__(self):
        return f"generasjon: {self._generasjon} - meloneeeeer: {len(self._aaker)}"

class Vannmelon: #han er en enkel liten gutt

    def __init__(self):
        self._ant_fro = 10

    def hent_ant_fro(self):
        return self._ant_fro
```

```python
#Testprogram:

gaard = Gaard()
neste = input("")

while neste != "x":
    gaard.plant_vannmelon()
    print(gaard)
    neste = input("")
```