3a

```
def endreBokstav(tekst):
    nyTekst = ""
    for bokstav in tekst:
        if bokstav.isupper():
            nyTekst += bokstav.lower()
        else:
            nyTekst += bokstav.upper()
    return nyTekst


assert endreBokstav("a") == "A"
assert endreBokstav("aaa") == "AAA"
assert endreBokstav("AAA") == "aaa"
assert endreBokstav("AbCd") == "aBcD"
```

6 points:

- Correct function definition and argument: 1p
- Use of loop 2p
- Successfully build up correct nyTekst string 2p
- Working solution incl return 1p

3b

```
# v1
gyldig_rangering(lag_a, lag_b, lag_c):
    return set(lag_a, lag_b, lag_c) == {1, 2, 3}

# v2
gyldig_rangering(lag_a, lag_b, lag_c):
    abc_set = set(lag_a, lag_b, lag_c)
    if 1 in abc_set:
        if 2 in abc_set:
            if 3 in abc_set:
                return True
    return False


assert gyldig_rangering(1, 2, 3) == True
assert gyldig_rangering(1, 3, 2) == True
assert gyldig_rangering(2, 1, 3) == True
assert gyldig_rangering(2, 3, 1) == True
assert gyldig_rangering(3, 1, 2) == True
assert gyldig_rangering(3, 2, 1) == True
assert gyldig_rangering(6, 0, 0) == False
assert gyldig_rangering(1, 2, 2) == False
```

6 points:

- Correct function definition and argument: 1p
- Logic testing if 1, 2, 3 are present among input args: 4p (any working solution gets full points, subtract as deemed appropriate for almost-working solutions)
- Working solution incl return 1p

3c

```
def garnkalkulator(genser_moenster, lengde_pr_noeste):
    antall_pr_farge = {}

    for farge in genser_moenster.keys():
        farge_trenges = genser_moenster[farge]
        farge_lengde = lengde_pr_noeste[farge]

        antall_noester = 0

        while farge_trenges > 0:
            farge_trenges -= farge_lengde
            antall_noester += 1

        antall_pr_farge[farge] = antall_noester

    return antall_pr_farge


assert garnkalkulator({"rød": 400}, {"rød": 150}) == {"rød": 3}
assert garnkalkulator({"rød": 400}, {"rød": 100}) == {"rød": 4}
assert garnkalkulator({"rød": 200, "blå": 800, "hvit": 70}, {"rød": 150,
"blå": 230, "hvit": 50}) == {"rød": 2, "blå": 4, "hvit": 2}
```

8 points:

- Correct use of parameters and return: 1p
- Looping over different colors: 2p
- Correct use of dictionaries (retrieve/add values): 2p
- Determining how much yarn is needed of a color: 2p
    - Almost correct: if >= 0 is used instead of > 0 (resulting in `antall_noester=5` when `farge_trenges=400` and `farge_lengde=100`): 1p
- Complete working solution: 1p

3d

```
# v1
def fremhev_fremdrift(alle_dager):
    hoeyere = []
    for dag in alle_dager:
        er_hoeyere = True
        for tidligere_dag in hoeyere:
            if tidligere_dag > dag:
                er_hoeyere = False
        if er_hoeyere:
            hoeyere.append(dag)

    return hoeyere

# v2
def fremhev_fremdrift(alle_dager):
    hoeyere = [alle_dager[0]]
    for dag in alle_dager[1:]:
        if dag >= hoeyere[-1]:
            hoeyere.append(dag)
    return hoeyere


assert fremhev_fremdrift([1, 2, 3]) == [1, 2, 3]
assert fremhev_fremdrift([1, 1]) == [1, 1]
```

```
assert fremhev_fremdrift([1, 3, 2, 4, 6, 5]) == [1, 3, 4, 6]
```

8 points:

- Correct use of parameters and return: 1p
- Looping over alle_dager: 2p
- Testing if dag is higher than previous days: 2p
- Correctly constructing a new list of days or correctly modifying alle_dager: 2p
- Complete working solution: 1p

3e

```
def matrise_til_sparsom(matrise):
    sparsom_matrise = {}

    for i in range(len(matrise)):
        for j in range(len(matrise[i])):
            verdi = matrise[i][j]
            if verdi > 0:
                if i in sparsom_matrise:
                    sparsom_matrise[i][j] = verdi
                else:
                    sparsom_matrise[i] = {j: verdi}

    return sparsom_matrise

matrise = [[ 0, 1.5, 3.2], [1.2, 0, 0], [ 0, 2.0, 1.3]]
sparsom_matrise = {0: {1: 1.5, 2: 3.2}, 1: {0: 1.2}, 2: {1: 2.0, 2: 1.3}}


assert matrise_til_sparsom(matrise) == sparsom_matrise
```

10 points:

- Correct use of parameters and return: 1p
- Using a double for loop: 2p
- Correct inner for loop (loop over matrise[i]): 1p
- Testing if value > 0: 1p
- Correctly adding value to sparsom_matrise: 3p
- Complete working solution: 2p

4a

```
class Ansatt:
    def __init__(self, navn, tittel):
        self.navn = navn
        self.tittel = tittel

    def hentNavn(self):
        return self.navn

    def hentTittel(self):
        return self.tittel

    def erKaptein(self):
        return self.tittel == "Kaptein"

    def erStyrmann(self):
        return self.tittel == "Styrmann"
```

6 points:

- Constructor: 3p
- hentNavn & hentTittel: 1p
- erKaptein & erStyrmann: 2p

4b

```
class Mannskap:
    def __init__(self, flightId):
        self.flightId = flightId
        self.piloter = []
        self.kabinansatte = []

    def hentFlightId(self):
        return self.flightId

    def tilordnePilot(self, nyAnsatt):
        if nyAnsatt.erKaptein() or nyAnsatt.erStyrmann():
            self.piloter.append(nyAnsatt)
            return True
        else:
            return False

    def tilordneKabinansatt(self, nyAnsatt):
        if nyAnsatt is None:
            return False
        else:
            self.kabinansatte.append(nyAnsatt)
            return True

    def erLovligMannskap(self, maxPax):
        # Minst to piloter
        if len(self.piloter) < 2:
            return False

        # en av pilotene må være Kaptein
        harKaptein = False
        for pilot in self.piloter:
            if pilot.erKaptein():
                harKaptein = True

        if not harKaptein:
            return False

        # minst en kabinansatt
        if len(self.kabinansatte) == 0:
            return False

        # minst en kabinansatt for hver 50ende passasjer
        minst_kabinansatte = 0
        while maxPax > 0:
            minst_kabinansatte += 1
            maxPax -= 50

        if len(self.kabinansatte) < minst_kabinansatte:
            return False

        return True
```

10 points:

- constructor, including empty lists for piloter & kabinansatte: 2p
- tilordnePilot & tilordneKabinansatt: 2p
- erLovligMannskap:
    - test 2 pilots: 1p,
    - test captain: 2p,
    - test at least one kabinansatt: 1p,
    - test one kabinansatt per 50 passengers: 2p
- -1p if hentFlightId is missing

4c

```
class Fly:
    def __init__(self, regId, maxPax):
        self.regId = regId
        self.maxPax = maxPax
        self.mannskap = None

    def hentRegId(self):
        return self.regId

    def hentMaxPax(self):
        return self.maxPax

    def hentMannskap(self):
        return self.mannskap

    def validerMannskap(self):
        if self.mannskap is not None:
            return self.mannskap.erLovligMannskap(self.maxPax)
        else:
            return False

    def tilordneMannskap(self, nyttMannskap):
        self.mannskap = nyttMannskap
```

7 points:

- Constructor, incl mannskap: 2p
- 'hent' functions: 1p
- validerMannskap:
    - consider self.mannskap can be none: 1p
    - correct calling of erLovligMannskap: 2p
- tilordneMannskap: 1p

4d

```
class Flyselskap:
    def __init__(self, navn):
        self.navn = navn
        self.fly = {}

    def hentNavn(self):
        return self.navn

    def leggTilFly(self, nyttFly):
        self.fly[nyttFly.regId] = nyttFly

    def finnFly(self, flyRegId):
        if flyRegId in self.fly:
```

```
            return self.fly[flyRegId]
        else: # made explicit here, but can be left out
            return None

    def tilordneMannskap(self, flyRegId, mannsk):
        fly = self.finnFly(flyRegId)
        if fly is not None:
            fly.tilordneMannskap(mannsk)
```

8 points:

- constructor, incl empty dictionary for fly: 2p
- leggTilFly: 2p
- finnFly:
  - use of dictionary 1p
  - test if flyRegId in dictionary: 1p
- tilordneMannskap: 2p
- -1 if hentNavn missing

4e

```
    def hentKopiAvFlyOrdbok(self):
        nyOrdbok = {}
        for regId in self.fly:
            nyOrdbok[regId] = self.fly[regId]

        return nyOrdbok
```

5 points:

- Successfully creating copy of self.fly: 3p
- Fully working solution: 2p

4f

```
def hovedprogram():
    sas = Flyselskap("SAS")

    fly = Fly("LN-RKF", 247)
    mannskap = Mannskap("SK935")
    mannskap.tilordnePilot(Ansatt("Torhild", "Kaptein"))
    mannskap.tilordnePilot(Ansatt("Marianne", "Styrmann"))
    mannskap.tilordneKabinansatt(Ansatt("Terje", "Purser"))
    mannskap.tilordneKabinansatt(Ansatt("Rigmor", "Flyvert"))
    mannskap.tilordneKabinansatt(Ansatt("Solveig", "Flyvert"))
    mannskap.tilordneKabinansatt(Ansatt("Jorolf", "Flyvert"))
    mannskap.tilordneKabinansatt(Ansatt("Norunn", "Flyvert"))

    sas.leggTilFly(fly)
    sas.tilordneMannskap("LN-RKF", mannskap)


    if sas.finnFly("LN-RKF").validerMannskap():
        print("Mannskapet er klart til take-off!")
    else:
        print("Det mangler noen ombord, vi må avbryte.")


hovedprogram()
```

7 points:

- Creating Flyselskap, Fly, Mannskap, Ansatt objects: 3p
- Correct use of leggTilFly: 1p
- Correct use of tilordneMannskap: 1p
- Test and print results of .validerMannskap: 2p