Forside

Oppgave	Tittel	Maks poeng	Oppgavetype
i	Informasjon		Informasjon eller ressurser

Del 1 (8 poeng)

Oppgave	Tittel	Maks poeng	Oppgavetype
1(a)	Oppgave 1a	1	Flervalg
1(b)	Oppgave 1b	1	Flervalg (flere svar)
1(c)	Oppgave 1c	1	Flervalg
1(d)	Oppgave 1d	1	Flervalg
1(e)	Ooppgave 1e	1	Flervalg
1(f)	Oppgave 1f	1	Flervalg
1(g)	Oppgave 1g	2	Flervalg (flere svar)

Del 2 (9 poeng)

Oppgave	Tittel	Maks poeng	Oppgavetype
2(a)	Oppgave 2a	2	Flervalg
2(b)	Oppgave 2b	2	Flervalg
2(c)	Oppgave 2c	2	Flervalg
2(d)	Oppgave 2d	3	Flervalg (flere svar)

Del 3 (27 poeng)

Oppgave	Tittel	Maks poeng	Oppgavetype

3(a)	Oppgave 3a	4	Programmering
3(b)	Oppgave 3b	6	Programmering
3(c)	Oppgave 3c	6	Programmering
3(d)	Oppgave 3d	4	Programmering
3(e)	Oppgave 3e	7	Programmering

Del 4 (47 poeng)

Oppgave	Tittel	Maks poeng	Oppgavetype
4	Oppgave 4	47	Programmering

Del 5 (9 poeng)

Oppgave	Tittel	Maks poeng	Oppgavetype
5(a)	Oppgave 5a	5	Programmering
5(b)	Oppgave 5b	4	Programmering

Informasjon

UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Skriftlig eksamen i IN1000
2023 høst
Varighet: 1.12.23, 9:00 til 1.12.23, 13:00
Tillatte hjelpemidler: Alle trykte og skrevne, ingen elektroniske

Det er viktig at du leser hele forsiden før du begynner å besvare eksamensoppgavene.

Om du ønsker å zoome i oppgavesettet, hold ctrl nede og trykk + eller - på numerisk tastatur.

Faglærere vil besøke lokalet ca kl 10-11.

Dersom du mener en programmeringsoppgave er uklar, kan du gjøre egne forutsetninger og beskrive disse.

Du kan veksle mellom ulike språk (bokmål eller engelsk) øverst på siden.

Oppgavene gir til sammen 100 poeng. Poengfordelingen sier noe om vektingen av deloppgaver i sensuren.

1(a) Oppgave 1a

```
1   a = "Hei"
2   b = "Hå"
3
4   b = a
5
6   print(a, b)
```

Hva vil koden over skrive ut?
Velg ett alternativ:

- Hei Hå
- Hei Hei ✔
- Hå Hei
- Hå Hå

Maks poeng: 1

1(b) Oppgave 1b

```
1   tall = 8
2
3   if tall < 4:
4       if tall > 7:
5           print("Linje 5 kjører!")
6   elif tall < 10:
7       print("Linje 7 kjører!")
8   elif tall > 100:
9       print("Linje 9 kjører!")
10  else:
11      print("Linje 11 kjører!")
```

Hvilke av kodelinjene med **print** kommer til å kjøre i dette programmet?
Velg ett eller flere alternativer

- [ ] Linje 11
- [ ] Linje 7 ✔
- [ ] Linje 9
- [ ] Linje 5

Maks poeng: 1

1(c) Oppgave 1c

```
1   # fyll inn her
2   while teller <= 10:
3       print(teller)
4       teller = teller + 1
```

Hva må fylles inn på linje 1 for at programmet skal skrive ut tallene fra og med 1 til og med 10?
Velg ett alternativ:

- teller = range(10)
- i = range(10)
- i = 0
- i = 1
- teller = 1 ✔
- teller = 0

Maks poeng: 1

1(d) Oppgave 1d

```
1    for i in range(10):
2    # fyll inn her
```

Hva må fylles inn på linje 2 for at programmet skal skrive ut alle tallene fra og med 1 til og med 10? (Legg merke til om det er innrykk eller ikke i alternativet du velger.)

Velg ett alternativ:

- print(i + 1) ✔
- print(i + 1)
- print(i)
- print(i)
- print(i = i + 1)
- print(i = i + 1)

Maks poeng: 1

1(e) Ooppgave 1e

```
1   from random import randint
2
3   terningkast = randint(1,6)
4   print(terningkast)
5   # fyll inn her
6       terningkast = randint(1,6)
7       print(terningkast)
```

I dette programmet gir funksjonskallet **randint(1,6)** oss et tilfeldig heltall fra og med 1 til og med 6, akkurat som et terningkast.

Hva må du skrive på linje 5 for at programmet skal fortsette å generere tilfeldige tall helt til du får en sekser?

Velg ett alternativ

- while terningkast == 6:
- while i < 6:
- while i == 6:
- for terningkast < 6:
- while terningkast < 6: ✔
- for i < 6:

Maks poeng: 1

1(f) Oppgave 1f

```
vers = 100
while vers > 0:
    print("Fiskebollen lever i havet")
    print("Havet er fiskebollens hjem")
    print("Dette er er vers nummer", vers)
    print("Nå er det bare", vers - 1, "igjen")
    print()
print("Sangen er slutt!")
```

Hva blir resultatet av å kjøre denne løkken?

Velg ett alternativ:

- ○ Uendelig mange vers skrives ut ✔
- ○ 100 vers skrives ut
- ○ 99 vers skrives ut
- ○ Ingen vers skrives ut

Maks poeng: 1

1(g) Oppgave 1g

Finn (kryss av for) ALLE feilene i dette programmet (det kan være mer enn en):

```
def superfunksjon(x):
    return "Den kuleste verdien er " + x

superfunksjon(42)
print(x)
```

Programmet får denne feilmeldingen når det prøver å kjøre:

```
Traceback (most recent call last):
  File "feil.py", line 15, in <module>
    superfunksjon(42)
  File "feil.py", line 13, in superfunksjon
    return "Den kuleste verdien er " + x
TypeError: can only concatenate str (not "int") to str
```

Velg ett eller flere alternativer

- [ ] Man må alltid bruke komma for å slå sammen strenger, det virker ikke med +
- [ ] Det er brukt **x** i stedet for **str(x)** i returverdien ✔
- [ ] Den lokale variabelen **x** er ikke definert utenfor funksjonen ✔
- [ ] Man har glemt å bruke parameteren **self** i funksjonen
- [ ] Det gir feilmelding hvis en funksjon med returverdi kalles uten at returverdien lagres i en variabel
- [ ] Lengden av strengen er kortere enn 42 tegn

Maks poeng: 2

2(a) Oppgave 2a

Denne koden er felles for alle deloppgavene på oppgave 2:

```
1   class Gås:
2       def __init__(self, navn):
3           self._navn = navn
4           self._kaffekopper = 0
5
6       def drikk_kaffe(self):
7           self._kaffekopper += 1
8           if self._kaffekopper < 2:
9               print(self._navn, "koser seg med en god kopp kaffe")
10          else:
11              print(self._navn, "drakk for mye kaffe og føler seg ikke bra!")
12
13  gås0 = Gås("Arthur")
14  gås1 = Gås("Baldur")
15  gås2 = Gås("Honk")
16  favorittgås = gås2
17
18  gås1 = gås0
19  gås1.drikk_kaffe()              # oppgave 2a
20  gås2 = gås1
21  gås2.drikk_kaffe()              # oppgave 2b
22  favorittgås.drikk_kaffe()       # oppgave 2c
23
```

Hva skrives ut på linje 19?

Velg ett alternativ:

- Honk drakk for mye kaffe og føler seg ikke bra!
- Baldur drakk for mye kaffe og føler seg ikke bra!
- Honk koser seg med en god kopp kaffe
- Arthur koser seg med en god kopp kaffe ✔
- Arthur drakk for mye kaffe og føler seg ikke bra!
- Baldur koser seg med en god kopp kaffe

Maks poeng: 2

2(b) Oppgave 2b

Denne koden er felles for alle deloppgavene på oppgave 2:

```
1   class Gås:
2       def __init__(self, navn):
3           self._navn = navn
4           self._kaffekopper = 0
5
6       def drikk_kaffe(self):
7           self._kaffekopper += 1
8           if self._kaffekopper < 2:
9               print(self._navn, "koser seg med en god kopp kaffe")
10          else:
11              print(self._navn, "drakk for mye kaffe og føler seg ikke bra!")
12
13  gås0 = Gås("Arthur")
14  gås1 = Gås("Baldur")
15  gås2 = Gås("Honk")
16  favorittgås = gås2
17
18  gås1 = gås0
19  gås1.drikk_kaffe()              # oppgave 2a
20  gås2 = gås1
21  gås2.drikk_kaffe()              # oppgave 2b
22  favorittgås.drikk_kaffe()       # oppgave 2c
23
```

Hva skrives ut på linje 21?

Velg ett alternativ:

- Arthur drakk for mye kaffe og føler seg ikke bra! ✔
- Honk koser seg med en god kopp kaffe
- Honk drakk for mye kaffe og føler seg ikke bra!
- Baldur drakk for mye kaffe og føler seg ikke bra!
- Arthur koser seg med en god kopp kaffe
- Baldur koser seg med en god kopp kaffe

Maks poeng: 2

2(c) Oppgave 2c

Denne koden er felles for alle deloppgavene på oppgave 2:

```
1   class Gås:
2       def __init__(self, navn):
3           self._navn = navn
4           self._kaffekopper = 0
5
6       def drikk_kaffe(self):
7           self._kaffekopper += 1
8           if self._kaffekopper < 2:
9               print(self._navn, "koser seg med en god kopp kaffe")
10          else:
11              print(self._navn, "drakk for mye kaffe og føler seg ikke bra!")
12
13  gås0 = Gås("Arthur")
14  gås1 = Gås("Baldur")
15  gås2 = Gås("Honk")
16  favorittgås = gås2
17
18  gås1 = gås0
19  gås1.drikk_kaffe()              # oppgave 2a
20  gås2 = gås1
21  gås2.drikk_kaffe()              # oppgave 2b
22  favorittgås.drikk_kaffe()       # oppgave 2c
23
```

Hva skrives ut på linje 22?

Velg ett alternativ:

- Honk koser seg med en god kopp kaffe ✔
- Arthur koser seg med en god kopp kaffe
- Baldur drakk for mye kaffe og føler seg ikke bra!
- Honk drakk for mye kaffe og føler seg ikke bra!
- Arthur drakk for mye kaffe og føler seg ikke bra!
- Baldur koser seg med en god kopp kaffe

Maks poeng: 2

2(d) Oppgave 2d

Denne koden er felles for alle deloppgavene på oppgave 2:

```
1   class Gås:
2       def __init__(self, navn):
3           self._navn = navn
4           self._kaffekopper = 0
5
6       def drikk_kaffe(self):
7           self._kaffekopper += 1
8           if self._kaffekopper < 2:
9               print(self._navn, "koser seg med en god kopp kaffe")
10          else:
11              print(self._navn, "drakk for mye kaffe og føler seg ikke bra!")
12
13  gås0 = Gås("Arthur")
14  gås1 = Gås("Baldur")
15  gås2 = Gås("Honk")
16  favorittgås = gås2
17
18  gås1 = gås0
19  gås1.drikk_kaffe()              # oppgave 2a
20  gås2 = gås1
21  gås2.drikk_kaffe()              # oppgave 2b
22  favorittgås.drikk_kaffe()       # oppgave 2c
23
```

Kryss av for alle påstandene som er sanne:
Velg ett eller flere alternativer

- [ ] Hvis vi legger til **gås0._kaffekopper = 0** i linje 23 bruker vi klassen slik den er ment å brukes
- [ ] Konstruktøren **__init__(navn)** er en del av grensesnittet til klassen **Gås** ✔
- [ ] Hvis vi legger til **self._navn = "Monsieur Trèsdangereux"** i linje 23 får vi en feilmelding når programmet skal kjøre ✔
- [ ] Metoden **drikk_kaffe()** er en del av grensesnittet til klassen **Gås** ✔
- [ ] Hvis vi erstatter **self._kaffekopper** med **self.kaffekopper** i alle metodene, vil denne variabelen ha en felles verdi for alle objekter av klassen **Gås**
- [ ] Variabelen **self._kaffekopper** er en del av grensesnittet til klassen **Gås**

Maks poeng: 3

3(a) Oppgave 3a

Write a function **points** that takes a string as a parameter. We will assume this string is one of *"firkløver"*, *"hjerte"*, *"lyn"* or *"lyspære"*. The function should return an integer equal to the number of letters in the parameter string - the exception being *"lyn"*, where the function should return 0.
Skriv ditt svar her

Maks poeng: 4

3(b) Oppgave 3b

Anta at du har en funksjon **poeng** som tar inn en streng med tekst som parameter og returner en poengsum (et heltall). Du må ikke ha løst deloppgave 3a for å kunne løse denne oppgaven - vi antar at funksjonen eksisterer og virker som den skal.

Lag en ny funksjon **poengsum** som tar inn en liste med strenger som parameter, og bruker funksjonen **poeng** til å regne ut hvor mange poeng hver streng er verdt. Funksjonen din skal returnere *summen* av poengene for alle strengene i listen.
Skriv ditt svar her

Maks poeng: 6

3(c) Oppgave 3c

Lag en funksjon **velg** som tar inn en liste som parameter. Funksjonen din skal be brukeren om å skrive inn et heltall (vi antar at brukeren aldri vil skrive inn noe annet):

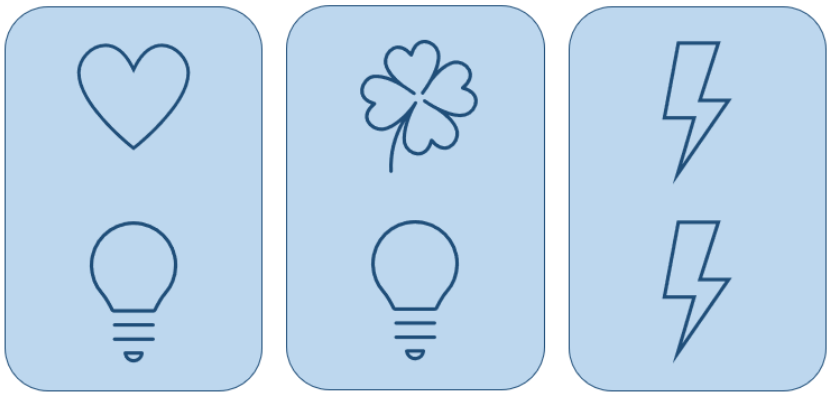
- Hvis heltallet ikke er en gyldig indeks i listen, skal funksjonen spørre brukeren om igjen helt til det kommer inn en gyldig indeks
- Hvis heltallet er en gyldig indeks i listen, skal funksjonen returnere elementet som ligger på denne indeksen

Du trenger ikke gi brukeren informasjon om hvilke tall som vil være gyldige valg.
Skriv ditt svar her

Maks poeng: 6

3(d) Oppgave 3d

Vi lager et spill med følgende spillkort, som hver er verdt et visst antall poeng når de velges av brukeren:



I spillet representerer vi disse tre kortene som tre lister med to strenger i hver, på denne måten:

```
kort1 = ["hjerte", "lyspære"]
kort2 = ["firkløver", "lyspære"]
kort3 = ["lyn", "lyn"]

kortliste = [kort1, kort2, kort3]
```

Lag en funksjon **spill_et_kort** som tar inn en slik **kortliste** som parameter. Funksjonen skal la brukeren velge ett av de tre kortene og skrive ut til skjermen hvor mange poeng brukeren fikk for kortet som ble valgt.

Du kan anta at følgende funksjoner eksisterer og virker som de skal, selv om du ikke har gjort de forrige deloppgavene:

- En funksjon **poengsum** som tar inn en liste med strenger (representerer et kort) som parameter og returnerer et heltall (som representerer poengsum for et kort)
- En funksjon **velg** som ber brukeren om å velge et element fra en liste ved å skrive inn et heltall (indeks i listen) og returnerer elementet på denne indeksen

Skriv ditt svar her

Maks poeng: 4

3(e) Oppgave 3e

```
ord = ["F", "E", "R", "S", "K", "E", "N", "B", "R", "U", "S"]
tell_tegn = {}
for tegn in ord:
    if tegn not in tell_tegn:
        tell_tegn[tegn] = 0
    tell_tegn[tegn] += 1
for tegn in tell_tegn:
    print("Antall", tegn, ":", tell_tegn[tegn])

setning = ["Flodhest", "er", "best", "ingen", "protest"]
tell_ord = {}
for ord in setning:
    if ord not in tell_ord:
        tell_ord[ord] = 0
    tell_ord[ord] += 1
for ord in tell_ord:
    print("Antall", ord, ":", tell_ord[ord])
```

Forenkle programmet over så mye som mulig ved hjelp av en prosedyre som skal kalles to ganger, og som skal kunne gjenbrukes senere på andre lister.

Du skal skrive både prosedyren og programmet som bruker den, og ditt program skal gi akkurat samme utskrift som programmet over.

Skriv ditt svar her

Maks poeng: 7

4 Oppgave 4

Oppgaveteksten til oppgave 4 ligger i vedlagte PDF dokument. Du kan legge til egne metoder i løsningen din ved behov, kommenter i så fall disse kort. Om du står fast på en deloppgave, fortsett på de som kommer etter. Du kan bruke metoder beskrevet i oppgaveteksten selv om du ikke selv skriver dem.

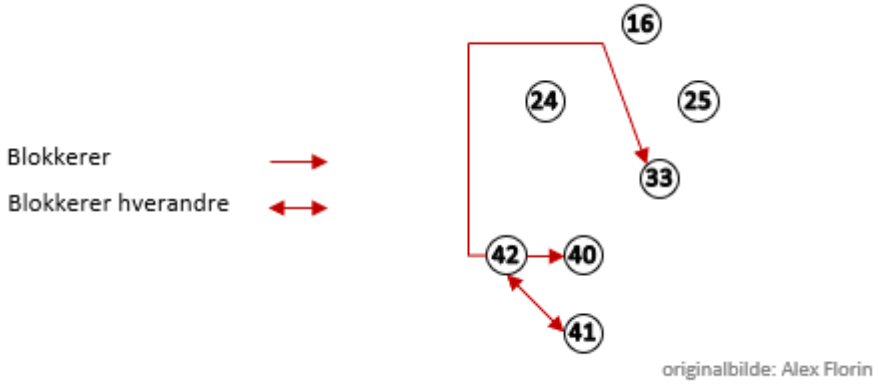
Skriv ditt svar her

Maks poeng: 47

5(a) Oppgave 5a

I brettspillet *Gloomhaven* kan man spille mange forskjellige oppdrag som har hvert sitt nummer. Når du fullfører et oppdrag, kan dette blokkere andre oppdrag (slik at disse ikke lenger kan spilles).

Et forenklet flytskjema for noen av oppdragene i spillet ser du her:



Dette representerer vi med en ordbok **oppdrag** hvor nøklene er oppdragsnummer, og verdiene lister hvor

- Element 0 viser om oppdraget er fullført (**True**) eller ikke (**False**)
- Element 1 er en liste over oppdrag som kan blokkere dette oppdraget hvis de fullføres

Før noen av oppdragene er fullført, vil denne ordboken dermed se slik ut:

```
#           oppdragsnummer: [fullført, blokkeres_av]
oppdrag = { 16: [False, []],
            24: [False, []],
            25: [False, []],
            33: [False, [42]],
            40: [False, [42]],
            41: [False, [42]],
            42: [False, [41]]}
```

Lag en funksjon **er_blokkert(nummer, oppdrag)** som tar inn følgende parametre:

- **nummer**: heltallsnummer på et oppdrag som vi skal sjekke om er blokkert
- **oppdrag:** en ordbok på formatet vist i eksempelet over, men ikke nødvendigvis med de samme verdiene

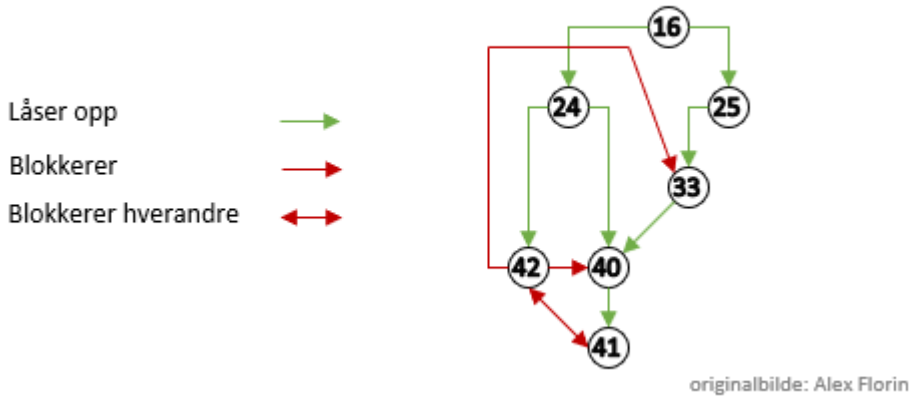
Funksjonen skal returnere **True** hvis oppdraget med nummer **nummer** er blokkert av et annet oppdrag som er fullført, og **False** hvis ikke.
I denne deloppgaven kan du anta at **nummer** alltid vil være en nøkkel som finnes i **oppdrag**.

Skriv ditt svar her

Maks poeng: 5

5(b) Oppgave 5b

I tillegg til blokkering, kreves det ofte at andre oppdrag er fullført for at et oppdrag skal låses opp og kunne spilles. Vi kan legge til dette i flytskjemaet, som da ser slik ut:



For å representere dette utvider vi ordboken **oppdrag** fra forrige deloppgave, slik at nøklene fortsatt er oppdragsnummer, og verdiene lister hvor

- Element 0 viser om oppdraget er fullført (**True**) eller ikke (**False**)
- Element 1 er en liste over oppdrag som kan blokkere dette oppdraget hvis de fullføres
- Element 2 er listen over oppdrag som alle må fullføres for å låse opp dette oppdraget

Før noen av oppdragene er fullført, vil denne ordboken se slik ut:

```
#           oppdragsnummer: [fullført, blokkeres_av, låses_opp_av]
oppdrag = { 16: [False, [], []],
            24: [False, [], [16]],
            25: [False, [], [16]],
            33: [False, [42], [25]],
            40: [False, [42], [24, 33]],
            41: [False, [42], [40]],
            42: [False, [41], [24]]}
```

Lag en funksjon **kan_spilles(nummer, oppdrag)** som tar inn følgende parametre:

- **nummer**: heltallsnummer på et oppdrag som vi skal sjekke om kan spilles
- **oppdrag:** en ordbok på formatet vist i eksempelet over, men ikke nødvendigvis med de samme verdiene

Funksjonen skal returnere **False** hvis oppdraget ikke kan spilles, det vil si hvis minst en av følgende stemmer:

- dette oppdragsnummeret finnes ikke i ordboken
- oppdraget er blokkert, det vil si at **er_blokkert(nummer, oppdrag)** returnerer **True** *(du kan anta at denne funksjonen virker selv om du ikke fikk til forrige deloppgave)*
- man har ikke fullført alle oppdragene som dette oppdraget låses opp av

On the other hand, if the scenario is playable, the function should return **True**.

Skriv ditt svar her

Maks poeng: 4

Question 17
Attached

Oppgave 4. Garderobe (47 poeng)

I denne oppgaven skal du hjelpe Kim å organisere garderoben sin. Kim har mange klesplagg og vil gjerne få skrevet et program som hjelper til med å sette sammen disse i nye antrekk.

En garderobe er i denne oppgaven samlingen av alle klærne til en person, mens en kategori er en type plagg. Eksempler på kategorier er bukser, skjorter, gensere, og t-skjorter. Alle bukser i garderoben hører for eksempel til i kategorien «bukser». Et plagg hører kun til i én kategori.

Et plagg har minst en farge, og flere plagg kan kombineres til et antrekk. Et antrekk består av plagg som har minst en felles farge, og passer til å brukes sammen i en eller flere anledninger (for eksempel tur eller selskap). Plaggene i et antrekk kommer fra hver sin kategori. For eksempel kan et antrekk bestå av en rød shorts (kategori «bukser») og en hvit t-skjorte (kategori «t-skjorter»). Et annet antrekk kan bestå av en blå langbukse (kategori «bukser»), en hvit skjorte (kategori «skjorter») og en grå og blå dressjakke (kategori «jakker»). Samme plagg kan være med i flere antrekk.

Du skal skrive klassene **Plagg, Kategori, Garderobe** og **Antrekk** til første versjon av et program til Kim:

- Klassen **Plagg** representerer enkeltplagg i garderoben.
- Klassen **Kategori** representerer en type plagg, og har oversikt over alle plaggene av denne typen i garderoben.
- Klassen **Antrekk** representerer et antrekk: Hvilke plagg som er med i antrekket, og hvilke(n) anledninger antrekket passer til.
- Klassen **Garderobe** organiserer alle plaggene inndelt i kategorier slik at hvert plagg hører til i én (og bare én) kategori. Dessuten har garderoben en liste over antrekk.

Instansvariabler skal ikke aksesseres utenfra klassen (de er non-public). Bruk metoder fra andre deloppgaver der dette er hensiktsmessig (selv om du ikke har løst dem). Du kan anta at returverdier og argumenter til metoder er gyldige, og trenger ikke teste for feilsituasjoner som ikke nevnes.

4a) 4p

Skriv klassen **Plagg** med konstruktør. Konstruktøren tar en parameter farger (liste av string). Klassen skal ha en instansvariabel for en liste av farger og en annen for hvor mange antrekk plagget er med i (int). Skriv også følgende metoder i klassen:

- **har_farge**. Tar parameteren farge (string). Metoden returnerer True hvis den oppgitte fargen er en av fargene i plagget, ellers False.
- **hent_ant_antrekk**. Returnerer antall antrekk plagget er med i.
- **oppdater_ant_antrekk**. Oppdaterer hvor mange antrekk plagget er med i. Parameter er endring i antallet (int, kan være negativ).

4b) 10p

Skriv klassen **Kategori** med konstruktør. Klassen skal ha instansvariabler for kategorinavn (string) og plagg (liste med referanser til plagg-objekter). Kategorinavnet er parameter til konstruktøren. Når kategorien opprettes har den ingen plagg. Videre skal du skrive følgende metoder i klassen:

- **nytt_plagg.** Parameter skal være farger (liste av string). Metoden oppretter et nytt objekt av klassen Plagg som legges til denne kategorien.
- **finn_plagg_med_farge**. Parameter farge (string). Returnerer liste med 0 eller flere referanser til Plagg-objekter med fargen.

- **trekk_tilfeldig_plagg**. Tar en farge som parameter og returnerer en referanse til et tilfeldig Plagg-objekt med riktig farge i kategorien. Om ingen plagg med fargen finnes i kategorien returneres None. Du kan bruke funksjonen **randint**(a, b) som returnerer et tilfeldig tall fra og med a, til og med b, for å trekke tilfeldig (du trenger ikke importere noe i denne besvarelsen).

4c) 7p Skriv klassen **Antrekk** med konstruktør. Et antrekk skal ha instansvariabler for anledninger (liste av strenger) og plagg (en liste med referanser til Plagg-objekter). Konstruktøren skal ta inn alle plaggene som er med i antrekket (liste av Plagg-referanser) og én anledning (string) som parametere, og oppdatere hvert av plaggene i antrekket for å registrere at de nå er med i et nytt antrekk.

Utvid klassen **Antrekk** med følgende metoder:

- **hent_plaggene**. Returnerer en liste med referanser til Plagg-objektene i antrekket.
- **legg_til_anledning**. Metoden legger til en anledning antrekket passer til. Anledning (string) tas inn som en parameter.
- **passer_til**. Metoden tar en anledning (string) som parameter og returnerer en boolsk verdi som sier om anledningen er en som dette antrekket passer til.
- **har_farge**. Metoden returnerer en boolsk verdi som sier om noen av plaggene har en bestemt farge. Fargen (string) er parameter.

4d) 7p Skriv klassen **Garderobe** med konstruktør. Klassen skal ha to instansvariabler:

- en ordbok der kategorinavn (string) er nøkkel og referanse til **Kategori**-objekt er verdi
- en liste med referanser til **Antrekk**-objekter

Ingen av disse initialiseres fra parametere til konstruktøren. Skriv også metoden:

- **nytt_plagg**. Metoden tar parametere kategorinavn (string) og farger (liste av string) og legger et nytt plagg inn i riktig kategori. Bruk en metode fra en tidligere deloppgave.

4e) 9p Skriv følgende metoder i klassen **Garderobe** som kan hjelpe Kim å finne passende antrekk:

- **finn_antrekk_til_anledning**. Metoden returnerer en liste med 0 eller flere referanser til alle Antrekk-objekter som passer til en anledning. Anledning (string) er parameter.
- **velg_første_antrekk**. Metoden returnerer en referanse til første antrekk den finner som passer til en bestemt anledning og der en bestemt farge finnes på minst et av plaggene. Anledning (string) og farge (string) er parametere til metoden. Finner den ikke et antrekk som passer returneres None.

4f) 10p Utvid klassen **Garderobe** med følgende metoder som lar Kim sette sammen nye antrekk:

- **finn_plagg_til_antrekk**. Metoden tar en liste med kategorinavn (liste av string) og en farge (string) som parametere. For hver kategori i listen med kategorinavn finner den et tilfeldig plagg med oppgitt farge, og returnerer disse plaggene (liste med referanser til Plagg-objekter). Hvis et av kategorinavnene ikke finnes i garderoben eller kategorien mangler plagg med oppgitt farge, returneres None.
- **lag_nytt_antrekk**. Metoden tar en liste med kategorinavn (liste av string), en farge (string) og en anledning (string) som parametere. Dersom den finner plagg med oppgitt farge for hver av kategorinavnene oppretter metoden et nytt antrekk med disse plaggene, legger det til garderoben og returnerer True. Finner den ikke plagg fra hver kategori returnerer metoden False.