Oppgave 1: Flervalgsoppgave (25%)

Bruk de to vedlagte svarskjemaene for å svare på denne oppgaven (ta vare på den ene selv). Du kan få et nytt ark av eksamensvaktene hvis du trenger det. **Kun et svar er helt riktig.** For hvert spørsmål gir korrekt avkryssing 1 poeng. Feil avkryssing, eller mer enn et kryss gir -½ poeng. Blankt svar gir 0 poeng. Du får ikke mindre enn 0 poeng totalt på denne oppgaven.

1) Vi har 1750 ulike tilstander som vi ønsker å representere. Hvor mange bit må vi minst bruke?

a) En byte (8 bit).
b) 11 bit.
c) 12 bit.
d) 2 byte (16 bit)

2) Anta at en RGB-farge angis med heksadesimale tall. Hvilken kode representerer en mørkegrå farge?

a) #FFFFFF
b) #404040
c) #506496
d) #300000

3) Hvor mye plass tar 20 minutter med (ukomprimert) stereo lyd av CD-kvalitet?

a) Omtrent 200 MB
b) Omtrent 500 MB
c) Omtrent 20 MB
d) Omtrent 1 GB

4) Anta IEEE floating-point representation av tall. Hvilken påstand er riktig?

a) Representasjonen består av tre deler: Fortegn, mantisse og eksponent.
b) Nøyaktigheten påvirkes av antall bits i mantissen.
c) Størrelsen på tallområdet som kan representeres påvirkes av antall bits i eksponenten.
d) Alle påstandene a-c er riktige.

5) Anta at et telefonnr (8 siffer) skal lagres. Hvilken representasjon tar minst plass?

a) Som heltall.
b) Som en streng av ASCII-tegn.
c) Som double.
d) Alternativene a-c tar like mye plass.

6) Hvert skritt i binærsøkalgoritmen

a) halverer søkerommet
b) finner søkenøkkelen
c) flytter et element
d) bytter 2 elementer

7) Hvilket av følgende er et krav på ei liste der vi vil bruke innstikksorteringsalgoritmen?

a) Listen må ha et odde antall elementer
b) Elementene må være sorterte
c) Det må finnes måter å fjerne og legge til elementer i listen
d) Ingen av disse kravene trenger å være oppfylt

8) Hva er den raskeste sikre måten å søke etter en enkelt verdi i en usortert tallrekke?

a) Skanne lineært gjennom alle elementene i rekken til verdien er funnet
b) Sortere rekken og utføre binærsøk
c) Velge tilfeldige elementer fra rekken til tallet er funnet
d) Det finnes ingen raskeste sikker måte

9) Verste fall i en lineær søkealgoritme oppstår når

a) det søkte elementet er et sted i midten av listen
b) det søkte elementet ikke er i listen i det hele tatt
c) det søkte elementet er det siste elementet i listen
d) det søkte elementet er det siste elementet i listen eller ikke er der i det hele tatt

10) Hvilken minneteknologi er raskest?

a) DDR-RAM
b) SSD
c) Cache
d) Alle disse er like raske

11) Hvordan virker monitoren?

a) Den viser tre forskjellige farger i hver piksel
b) Den blander fargene Rød, Gul og Blå for å lage alle mulige farger
c) Den regulerer lysstyrken avhengig av frekvensen på signalene fra maskinen
d) Alle alternativene er riktige

12) Hva er sant angående primær- og sekundærminne?

a) Primærminnet er permanent (ikke-flyktig)
b) Sekundærminnet kalles ofte for RAM
c) Primærminnet er mye større enn sekundærminnet
d) Ingen av de andre alternativene er riktig

13) Hva er sant angående datamaskiners historiske ytelses-forbedringer

a) Maskinen kan gå fortere jo tettere transistorene i hver integrerte krets er
b) Miniatyrisering gjør at klokkefrekvensen kan være over 1 GHz
c) Moores lov sier at antall transistorer på et areal dobles hvert andre år
d) Alle alternativene er riktige

14) Hvilke fem typer hoved-kretser finnes i Prosessoren (CPU)?

a) Instruksjon-hent (IF), Inst.-dekod (ID), Data-hent (DF), utfør (EX), Resultat-retur (RR)
b) Kontrollenhet, Aritmetisk-logisk enhet (ALU), Register, Input- og Outputkretser
c) Ingen av de andre alternativene er riktig
d) BIOS, ROM, Primærminne (RAM), Sekundærminne, Cache

15) Hva er en protokoll ?

a) Regler for hva en payload i en IP pakke kan inneholde
b) En beskrivelse av hvor raskt en melding kan overføres i et pakkesvitsjet nett som Internett
c) Regelverk som bestemmer hvordan kommunikasjon skal foregå og hvilke funksjoner som kan brukes
d) En oversikt over hvem som deltar i kommunikasjon på Internet

16) Hvilken oppgave har TCP protokollen som brukes på Internett ?

a) Tildeling av IP adresse, nettmaske og default gateway
b) Tilby logiske forbindelser og multipleksing av disse
c) Feilkorrigerende koding
d) Paritet, CRC eller Hash funksjoner

17) Dersom man ofte opplever at en tjeneste ikke virker når den ønskes benyttet, så beskrives dette som:

a) Dårlig ytelse på din forbindelse til Internett
b) Lav tilgjengelighet for den aktuelle tjenesten
c) Lav tiltro til den aktuelle tjenesten
d) Ustabil eller falsk DNS funksjon

18) Hvilke aspekter beskriver best de tekniske egenskapene ved en aksessteknologi ?

a) Kapasitet, Markedsandel og Prismodell
b) Protokoller, Installasjon og Terminalutstyr
c) Fleksibilitet, Pris og Bruksmønster
d) Kvalitet, Kapasitet og Effektivitet

19) Hvordan kan man oppdage om en melding har blitt endret underveis fra sender til mottaker ?

a) Ved å benytte analog signatur
b) Ved å benytte IPv6 i stedet for IPv4
c) Ved å benytte funksjoner som kan brukes av mottaker til å verifisere integriteten til meldingen
d) Ved å benytte funksjoner for å bevare konfidensialiteten til meldingen

20) Hvorfor benyttes ofte CRC for å detektere feil i digitale signaler ?

a) Fordi CRC har gode egenskaper med tanke på å oppdage burstfeil
b) Fordi CRC er bedre enn paritet og enkel sjekksum, samt like bra som hash funksjoner
c) På grunn av at CRC er veldig enkelt og effektivt
d) CRC har bra støtte i standardiserte protokoller

Oppgave 2: Grunnleggende programmering (20%)

I et parti sjakk belønnes vinneren med 1 poeng, taperen får 0 poeng, og ved remis (uavgjort) får begge ½ poeng hver. En sjakk-kamp spilles i et på forhånd bestemt antall partier, n. Trondheim sjakkforening (TSF) skal arrangere en kamp mellom de to stormestrene Carl Magnøssen (spiller nr. 1) og Sjakkma Ghandi (spiller nr. 2). TSF trenger din hjelp til å lage et program for å administrere kampen. I stedet for navnene til spillerne brukes kun numrene (1 og 2).

Oppgave 2a) (6%)

Lag funksjonen `chess_match()` som beskrives av følgende pseudokode:

```
procedure chess_match()
      Sett total_score1 ⟵ 0 # Totalpoeng til spiller 1
      Sett total_score2 ⟵ 0 # Totalpoeng til spiller 2

      Spør brukeren om hvor mange partier som skal spilles i kampen
      Sett num_games ⟵ antall partier

      Hvis brukeren gir et tall<1, skriv ut "Så kjedelig, da blir det ingen kamp!"
      Ellers, så lenge det er partier igjen å spille:
          Skriv ut "Parti" og nummeret på partiet
          Spør brukeren om antall poeng til spiller 1 i partiet
          Sett score1 ⟵ antall poeng til spiller 1 i partiet
          Spør brukeren om antall poeng til spiller 2 i partiet
          Sett score2 ⟵ antall poeng til spiller 2 i partiet
          Sett total_score1 ⟵ total_score1 + score1
          Sett total_score2 ⟵ total_score2 + score2

      Skriv ut "Kampen er slutt!"
      Skriv ut "Spiller 1 fikk " fulgt av totalpoengene til spiller 1 og "poeng."
      Skriv ut "Spiller 2 fikk " fulgt av totalpoengene til spiller 2 og "poeng."
```

Oppgaven skal teste om man kan oversette en pseudokode til standard Python.

```
def chess_match():
  total_score1 = 0  # totalpoeng til spiller 1
  total_score2 = 0  # totalpoeng til spiller 2

  num_games = int(input('Hvor mange partier skal spilles i kampen? '))

  if num_games < 1:
    print('Så kjedelig, da blir det ingen kamp.')
  else:
    for game in range(1, num_games+1):
      print("Parti ", game)
      score1 = float(input('Poeng for spiller 1:'))
      score2 = float(input('Poeng for spiller 2:'))
      total_score1 += score1
      total_score2 += score2

  print('Kampen er slutt!')
  print('Spiller 1 fikk', total_score1, 'poeng.')
  print('Spiller 2 fikk', total score2, 'poeng.')
```

```
def chess_match():
  total_score1 = 0
  total_score2 = 0

  num_games = int(input('Hvor mange partier skal spilles i kampen? '))

  if num_games < 1:
    print('Så kjedelig, da blir det ingen kamp.')
  else:
    game = 1
    while game <= num_games:
      print('Parti', game)
      score1 = float(input('Poeng for spiller 1:'))
      score2 = float(input('Poeng for spiller 2:'))
      total_score1 += score1
      total_score2 += score2
      game += 1

  print('Kampen er slutt!')
  print('Spiller 1 fikk',total_score1,'poeng.')
  print('Spiller 2 fikk',total_score2,'poeng.')
```

Oppgave 2b (3%)

Den spilleren som oppnår mer enn halvparten av de mulige poengene (dvs har n/2+0.5 eller fler poeng hvis kampen er inntil n partier) vinner kampen - da trenger ikke de gjenstående partiene å spilles. Hvis alle *n* partier er blitt spilt og de to spillerne har like mange poeng, slutter kampen uavgjort og man må spille ekstrapartier for å kåre en vinner. Hvis kampen er inntil 12 partier, kan den ende 6-6 med ekstraparti, eller ved at en av spillerne oppnår 6.5 eller 7 poeng (etter 7-12 partier).

Lag funksjonen

```
end_of_match(num_games, game, total_score1, total_score2)
```

som sjekker om kampen er slutt og som rapporterer om hvem som i så fall vant den. Funksjonen må altså sjekke om totalpoengene for en spiller er så høye at spilleren har vunnet kampen. Funksjonen tar 4 argumenter, to heltall (`num_games` og `game`) og to flyttall (`total_Score1` og `total_score2`), og returnerer enten 0 hvis kampen fortsatt pågår, nummeret til den spilleren som har vunnet kampen (1 eller 2) hvis kampen er avgjort, og 3 hvis kampen sluttet uavgjort.

Kommentar:

Oppgaven skal teste om man kan sette opp en standard valgstruktur i Python

Forslag:

```
def end_of_match(num_games,game,total_score1,total_score2):
    if total_score1 > num_games/2:
        return 1
    elif total_score2 > num_games/2:
        return 2
    elif game == num_games:
        return 3
    else:
        return 0
```

Oppgave 2c (5%):

I stedet for å spørre brukeren om antall poeng til spiller 2 i et parti, kan vi benytte at vi vet poengene for spiller 1, og at poengene til spiller 2 er avhengig av denne.

Lag funksjonen `chess_scorer()`.

Funksjonen skal spørre brukeren om resultatet for en spiller i et parti (dvs. 1, 0.5 eller 0) og returnere dette sammen med resultatet for motstanderen i det partiet (dvs. tilsvarende resultat: 0, 0.5 eller 1). Hvis brukeren oppgir et ugyldig resultat, skal funksjonen skrive ut "Umulig resultat" og spørre igjen.

Kommentar:

Oppgaven skal teste om man kan sette opp en standard løkkestruktur i Python. Oppgaven spør ikke om annen feilhåndtering enn å teste om et (tall)resultat er mulig eller ikke, så det er ikke nødvendig å sette opp en struktur for å håndtere om noen skriver noe annet enn et tall.:

```
def chess_scorer():
    score1 = float(input('Oppgi poeng for spiller nr. 1: '))
    while score1 not in (0,0.5,1):
        print("Umulig resultat")
        score1 = float(input('Oppgi poeng for spiller nr. 1: '))
    return score1, 1-score1
```

Oppgave 2d (6%):

Programmet i oppgave 2a ser bare på totalpoengene til en spiller, men lagrer ikke resultatene parti for parti. Anta at vi i stedet vil lagre alle resultatene til en spiller i ei liste og ha muligheten å hente ut totalpoengene til spilleren fra lista.

Lag funksjonen
`player_score(results)`.

Funksjonen skal ta inn som argument ei liste med resultat fra alle spilte partier for en spiller og returnere spillerens totalpoeng så langt i kampen (som et flyttall).
Listen i argumentet `results` er like lang som det antall partiert som skal spilles i kampen. Elementene i listen kan ha 4 forskjellige verdier: de tre mulige resultatene i et sjakkparti (0, 0.5, 1) og verdien `None` som tilsvarer at det partiet i kampen ikke er spilt enda. (Husk at datatypen til `None` er `NoneType`, og ikke f.eks. `float` som de andre verdiene i listen).

Kommentar:

Oppgaven skal teste om man kan sette opp en standard for-løkke i Python

```
def player_score(results):
    total_score = 0.0
    for n in results:
        if n != None:
            total_score += n
    return total_score

# NB: since None has its own datatype (NoneType),
# it is NOT possible to use the list method sum.
# def player_score(results):
#     return sum(results)
```

```
def player_score(results):
    total_score = 0.0
    for n in range(len(results)):
        if results[n] != None:
            total_score += results[n]
    return total_score
```

Oppgave 3a:

a,b = secret1(11,3)
a = 3
b = 2

secret1 beregner heltallsdivisjon (a) og resten (b), når argument 1 deles på argument 2.

Oppgave 3b:

m = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
answer = secret2(m)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Konverteres til

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

secret2 transponerer input-matrise m hvis den er kvadratisk, og returnerer -1 ellers.

NB: Det forutsettes ikke at studentene kan noe om matriser, så hvis de ikke bruker ordene matrise, kvadratisk eller transponere spiller ikke det noen rolle.

Oppgave 3c:

answer = secret3('148')

000101001000

secret3 konverterer en hexadesimaltall-streng til binærtall-streng (bortsett fra at hex siffer 2 skulle vært 0010 i oppgaveteksten!)

NB: Her var det 2 skrivefeil i oppgaveteksten for Python - feil i binærkoden for 2 (spilte ingen rolle i oppgaven), og det sto elif der det skulle stå else (som ville gitt syntaks-feil ved kjøring). Det ble opplyst om dette i lokalet.

Oppgave 4: Mer programmering (40%)

UKA trenger et system for å styre billettsalget. Du har meldt deg frivillig til å hjelpe til. (Hvis du ikke klarer å løse en deloppgave kan du likevel bruke funksjoner fra tidligere deloppgaver som om de er riktig implementert.) Det kan være lurt å kommentere koden.

Oppgave 4a (5%)

Lag en funksjon, `payment`, som tar inn billettpris og antall billetter og returnerer hvor mye kunden skal betale. Hvis man har kjøpt mer enn 3 billetter skal man få 10% rabatt på alle billettene.

Kommentar:

Oppgaven er en del av en helhet som utvikles gjennom hele oppgave 4. Den krever bare at man kan kalle en funksjon med parametre, gjøre en beregning basert på en enkel valgstruktur og vet hvordan man returnerer en verdi fra en funksjon.

```
def payment(ticket_price, number_of_tickets):
    if number_of_tickets > 3:
        return number_of_tickets*ticket_price*0.9
    else:
        return number_of_tickets*ticket_price
```

Oppgave 4b (5%)

Anta at det finnes en tekstfil, `prices.txt`, som inneholder konsertnavn og billettpris for konserten. Hver konsertoppføring er lagret på en linje i filen, der konsertnavn står først og pris kommer etter konsertnavnet separert med et semikolon (;).

Skriv en funksjon, `get_price`, som tar inn et konsertnavn og returnerer prisen for denne konserten. (Du må ta hensyn til tilfellet der konsernavnet ikke finnes!). Hvis konserten ikke finnes returnerer funksjonen prisen -1.

Eksempel på filinnhold:

```
The Rectorats;100
Gloshaugkameratene;150
The aller beste;250
```

Kommentar:

Oppgaven er en del av en helhet som utvikles gjennom hele oppgave 4. Den krever at man kan åpne (og lukke) en fil og vet hvordan man kan gjennomløpe et filinnhold med en løkkestruktur. Ut over dette krever den kun tekst- og listefunksjoner som finnes i Appendix.DSiden fil-funksjonene ikke var lagt med i Appendix, vild det ikke trekkes noe hvis syntax i open (og evt. close) setningene ikke er helt korrekt. Det er mange mulige løsninger - ingen er best - her er tre alternativer:

```
def get_price(concert_name):
    ticket_price = -1
    price_file = open('prices.txt','r')

    concert_line = price_file.readline()
    while concert_line != '' and ticket_price == -1:
        concert = concert_line.split(';')
        if concert[0] == concert_name:
            ticket_price = float(concert[1].rstrip('\n'))
            # Since each line in a text file ends with a newline (\n)
            # that character should be removed. However, overlooking
            # this in the exam will not lead to any point reduction.
        else:
            concert_line = price_file.readline()

    price_file.close()
    return ticket_price
```

```
def get_price(concert_name):
    ticket_price = -1
    price_file = open('prices.txt','r')

    for concert_line in price_file:
       concert = concert_line.split(';')
       if concert[0] == concert_name:
           ticket_price = float(concert[1].rstrip('\n'))

    price_file.close()
    return ticket_price
```

```
def get_price(konsertnavn):
    funnet = False
    with open('prices.txt', 'r') as fil:
        for linje in fil:
            if linje.find(konsertnavn) != -1:
                funnet = True
                break
    if funnet:
        liste = linje.split(';')
        return liste[1]
    else:
        return 0

# Her behøver man ikke eksplisitt lukke filen da dette
# skjer automatisk når with-strukturen er ferdig
```

Oppgave 4c (5%)

Lag en funksjon, `ticket`, som tar inn kjøpers navn, konsertnavn og antall billetter som argumenter. Bruk funksjonen i 4a til å generere pris som skal brukes i billett-teksten denne funksjonen skal generere. Billetten skal inneholde kjøpers navn, hvilken konsert, antall og totalpris. Billettprisen for konserten skal hentes fra filen `prices.txt` som ble brukt i oppgave 4b. Bruk funksjonene du skrev i 4a og 4b i denne oppgaven! (Hvis du ikke har løst 4a og 4b, kan du forutsette at de funksjonene finnes).

Eksempel på utskrevet billett:

```
****************************************
Uka 2015
****************************************
                  Navn:        Nils Nilsen
               Konsert:     The Rectorats
       Antall billetter:                8
             Totalpris:           720 kr
```

Kommentar:

Oppgaven er en del av en helhet som
utvikles gjennom hele oppgave 4. Denne oppgaven spør kun om å lage en pen utskrift, og tester om man kan kalle tidligere funksjoner for å hente verdier man trenger for dette. I tillegg må man bruke strengmetoder for å lage en pen utskrift. Oppgaven spør ikke om at funksjonen skal ha noen returverdier.

```
def ticket(buyer_name, concert_name, number_of_tickets):
    KOL1 = 18
    KOL2 = 20
    number_of_stars = KOL1 + KOL2 + 2

    # No points will be deducted for assuming that the concert exists,
    # that is, for assuming that get_price() does not return -1.

    ticket_price = get_price(concert_name)
    total_price = int(payment(ticket_price,number_of_tickets))
    stars = '*'*number_of_stars
    print(stars)
    print('Uka 2015')
    print(stars)
    print('Navn:'.rjust(KOL1),buyer_name.rjust(KOL2))
    print('Konsert:'.rjust(KOL1),concert_name.rjust(KOL2))
    print('Antall billetter:'.rjust(KOL1),str(number_of_tickets).rjust(KOL2))
    print('Totalpris:'.rjust(KOL1),str(total_price).rjust(KOL2),' kr')
    print(stars)

    # The ticket should be printed in a sensible and readable format.
    # No points will be deducted for not formatting it exactly as in
    # the example shown in the exam.
    # No points deducted for not using constants to represent fixed columns
```

Oppgave 4d (10%)

Lag en funksjon, `write_to_file`, som får billettinformasjon fra funksjonen i oppgave 4b: (navn, konsertnavn og antall billetter) og lagrer denne til en fil (`concerts.txt`). Filen skal inneholde 1 linje for hver billettransaksjon. Linjene skal bestå av konsertnavn, antall billetter, totalpris og kundenavn. Hvert element på linjen skal være skilt med et semikolon (;). Filnavnet skal være med som innparameter til funksjonen. Filen skal oppdateres underveis og skal ikke slettes hver gang den åpnes.
Eksempel på filinnhold:

```
The Rectorats;8;720;Nils Nilsen
Gloshaugkameratene;4;540;Per Persen
The Rectorats;2;200;Nina Karlsson
The aller beste;4;900;Even Evenrud
```

Kommentar:

Oppgaven er en del av en helhet som utvikles gjennom hele oppgave 4. Denne oppgaven dreier seg om å skrive til fil, men den informasjonen man trenger må beregnes av de andre funksjonene man har skrevet tidligere i oppgaven. Man trenger funksjonene fra opgave 4a og 4b. Informasjonen i parentes etter 4b: i oppgaveteksten kunne misforstås - den er der for å opplyse om hvilken informasjon funksjonen trenger for å fungere og har ikke noe med funksjonen get_price å gjøre. Siden dette kan virke forvirrende vil det ikke legges vekt på hvordan denne informasjonen kommer inn til funksjonen write_to_file.
Det sentrale i denne oppgaven er å hente informasjon fra andre funksjoner, sette denne sammen til en fornuftig linje, og skrive denne linjen til filen. Viktig å åpne og lukke filen, samt å åpne filen i modus 'a'.

```
def write_to_file(buyer_name, concert_name, number_of_tickets, file_name):
    ticket_price = get_price(concert_name)
    total_price = int(payment(ticket_price, number_of_tickets))
    concert_file = open(file_name,'a')
    concert_file.write(concert_name + ';' + str(number_of_tickets) \
                       + ';' + str(total_price) + ';' + buyer_name + '\n')
    concert_file.close()
```

```
def write_to_file(buyer_name, concert_name, number_of_tickets, file_name):
    ticket_price = get_price(concert_name)
    total_price = int(payment(ticket_price,number_of_tickets))
    line = concert_name + ';' + str(number_of_tickets) + ';' \
            + str(total_price) + ';' + buyer_name + '\n'
    with open(file_name,'a') as concert_file:
        concert_file.write(line)

    # Since each line in a text file ends with a newline (\n)
    # that character should be added. However, overlooking
    # this in the exam will not lead to any point reduction.
    # No points will be deducted for assuming that the concert exists,
    # that is, for assuming that get_price() does not return -1.
```

Oppgave 4e (15%)

Lag et menystyrt program som lar deg hente fra filen `concerts.txt` hvor mange billetter som er solgt til en gitt konsert, hvor stort beløp en gitt konsert har innbrakt, og totalinntekt for hele arrangementet.

Kommentar:

Det er mange måter å løse denne oppgaven på. Under er flere eksempler. I en er det valgt å ha 3 forskjellige funksjoner - et for hvert valg - og legge valget av funksjon i hovedfunksjonen. En annen mulighet er også vist. Programmet bør ha en sløyfe der et av valgene er om man skal avslutte.

```
def finn_antall_billetter(konsert):
    antall_billetter = 0
    with open('concerts.txt', mode = 'r') as fil:
        for linje in fil:
            liste = linje.split(';')
            if liste[0] == konsert:
                antall_billetter += float(liste[1])
    print('Det er solgt',antall_billetter,'billetter til konserten
med',konsert)

def finn_inntekt_for_konsert(konsert):
    inntekt = 0
    with open('concerts.txt', mode = 'r') as fil:
        for linje in fil:
            liste = linje.split(';')
            if liste[0] == konsert:
                inntekt += float(liste[2])
    print('Inntekten for konserten med',konsert,' er', inntekt)

def finn_totalinntekter():
    inntekt = 0
    with open('concerts.txt', mode = 'r') as fil:
        for linje in fil:
            liste = linje.split(';')
            inntekt += float(liste[2])
    print('Den totale inntekten for UKA 2015 er', inntekt)

def meny():
    stjerne = '*'*20
    print(stjerne)
    print('Menyprogram for UKA 2015')
    print('0: Avslutt')
    print('1: Antall billetter til en konsert')
    print('2: Inntekter for en konsert')
    print('3: Totalinntekter')
    print(stjerne)
    valg = 100
    tekst = 'Hvilken konsert:'
    while valg != 0:
        valg = int(input('Hva vil du gjøre:'))
        if valg == 0:
            print('Takk for nå')
        elif valg == 1:
            konsert = input(tekst)
            finn_antall_billetter(konsert)
        elif valg == 2:
            konsert = input(tekst)
            finn_inntekt_for_konsert(konsert)
        elif valg == 3:
            finn_totalinntekter()
        else:
            print('Du må skrive 0, 1, 2 eller 3')

meny()
```

```
def read_concert_file(option, concert_name):
    sum = 0
    concert_file = open('filtest.txt', 'r')
    for concert_line in concert_file:
        concert_item = concert_line.split(';')
        if option == 1 or option == 2:
            if concert_item[0] == concert_name:
                sum += int(concert_item[option])
                # concert_item[1] is the number of tickets
                # concert_item[2] is the amount generated
        else:
            sum += int(concert_item[2])
            # total amount generated
    concert_file.close()
    return sum

def main():
    option = -1
    print("***** Menu for tickets sold to Uka 2015. *****")
    print(">> Alternatives:")
    print(">> 0 = end")
    print(">> 1 = number of tickets sold to a given concert")
    print(">> 2 = amount generated by a given concert")
    print(">> 3 = total income for the whole arrangement")
    while option != 0:
        option = int(input(">> Select an option (0-3): "))
        if option == 1 or option == 2:
            concert_name = input(">> For which concert? ")
            if option == 1:
                print(">>", read_concert_file(option, concert_name), \
                        "tickets sold")
            else:
                print(">>", read_concert_file(option, concert_name), \
                        "kr generated")
        elif option == 3:
            print(">> Total income:", read_concert_file(option, ''), "kr")
```