Ordinær eksamen i Introduktion til programmering, blok 1, 2012

7. november 2012
Version 1.2

Dette dokument udgør opgavesættet for den ordinære eksamen i kurset "Introduktion til programmering", blok 1, 2012. Det består af 12 nummererede sider.

Dokumentet offentliggøres onsdag den 7. november kl. 9:00 på kursets hjemmeside via KU's kursusadministrationssystem Absalon. Besvarelsen skal afleveres senest fredag den 9. november kl. 9:00 — se dog afsnit 2 nedenfor hvis du sideløbende er indskrevet på kurset MatIntro eller et andet kursus med eksamen d. 7. eller 8. november.

Besvarelsen bedømmes efter 7-trinsskalaen ud fra en samlet bedømmelse af, hvorvidt læringsmålene for kurset er opfyldt (se kursusbeskrivelsen).

Eksamensresultaterne vil findes på Absalon senest tre uger efter eksamens afslutning og vil findes i KU's studieadministrative system kort tid derefter.

Eksamenssættet består af 8 opgaver. For alle opgaver kræves sigende kommentarer (herunder begrundelser for eller forklaringer af løsninger i det omfang, det ikke allerede fremgår af koden) og variabelnavngivning, god programmeringsstil og generel læsbarhed, herunder ved passende indrykning.

Opgavernes rækkefølge i sættet er uafhængig af deres sværhedsgrad. Læs hele sættet grundigt igennem, før du begynder at programmere.

Bemærk, at delvist færdige løsninger til enkeltopgaver kan give point.

Opgavesættet er berammet til at kunne løses korrekt med 16 timers koncentreret arbejdsindsats af studerende, som har kvalificeret sig til eksamen ved at løse de nødvendige obligatoriske opgaver.

Hvis der opstår tvivl om selvstændighed eller opnåelse af læringsmålene ud fra besvarelsen, kan studerende blive indkaldt til en supplerende mundtlig eksamen fredag den 23. november 2012, kl. 13. Studerende vil blive indkaldt til mundtlig eksamen med elektronisk brev til deres KU-konti senest torsdag den 22. november 2012 kl. 12:00, hvorfor *alle studerende skal efterse deres konto den 22. november 2012 om eftermiddagen.* Udeblivelse fra mundtlig eksamen vil resultere i indberettelse til studienævnet.

I tilfælde af uklarheder i opgaveteksten er det op til eksaminanden selv at specificere løsningens forudsætninger; se også afsnit 3 nedenfor.

1 Selvstændighed i besvarelsen og eksamenssnyd

Opgavesættet skal besvares *individuelt.* Det er kun tilladt at stille opklarende spørgsmål til opgavernes fortolkning til instruktorerne under instruktorvagtordningen og på kursets elektroniske diskussionsforum. Spørgsmål på forummet må ikke indeholde programtekst. Det er *ikke* tilladt for studerende på kurset at *besvare* spørgsmål om opgaverne på kurset forum – det er udelukkende instruktorer og undervisere, der må det.

Det er ikke tilladt at diskutere opgavernes fortolkning eller indholde med andre end de herover nævnte.

Det er *ikke* tilladt *overhovedet* at diskutere *besvarelse* af opgaverne, herunder afprøvningstilfælde, løsningsmetoder, algoritmer eller konkret programtekst. Specifikt er følgende *ikke* tilladt i eksamensperioden, og *enhver* overtrædelse vil resultere i indkaldelse til mundtlig overhøring samt overdragelse af sagen til studielederen til behandling under gældende regler for eksamenssnyd:

- At vise enhver del af sin besvarelse til andre, herunder specielt personer, som følger kurset.
- At vise enhver del af opgavesættet til personer, som ikke er tilknyttet kurset. Herunder at lægge (dele af) opgaveformuleringer online (fora og chatrooms inklusive) andetsteds end kursets diskussionsforum.
- At diskutere opgavesættet eller dets fortolkning med andre personer, udover de herover nævnte undtagelser.
- At efterlade opgavesættet eller noter/kladder til ens egen besvarelse uden opsyn på DIKU. Dette inkluderer at gå fra en ulåst PC selv i kortere tid, smide udskrifter i papirkurve på DIKU, eller udskrive opgavesæt og løsninger på DIKUs printere, da andre har adgang til disse. Vi kan ikke kontrollere en sådan adfærd, men hvis en anden persons aflevering ligner din, så vil I begge blive betragtet som ansvarlige for kopieringen.
- At diskutere med andre eller afskrive fra andre dele eller hele besvarelser af opgaver fra eksamenssættet.
- At efterlyse løsninger fra andre.
- At bruge i øvrigt tilladeligt skriftligt eller mundtligt materiale ud over kursets undervisningsmateriale *uden* henvisning til kilden (f.eks. oplysninger fra Wikipedia, Google Scholar eller lignende).

Brugen af *skriftligt* materiale fra offentligt tilgængelige kilder er *tilladt* under forudsætning af, at kilden angives i besvarelsen. For kilder på Internettet skal komplet URL angives.

Det indskærpes, at *alle* besvarelser vil blive underlagt både elektronisk og menneskelig plagiatkontrol. Denne plagiatkontrol er stærk nok til at genkende ligheder i kode selv om der er lavet forsøg på at skjule denne lighed med betydningsbevarende omkrivninger i koden.

Hvis eksaminatorer, studieleder og dekan finder det *sandsynligt*, at disse regler er overtrådt, så er konsekvensen som minimum, at eksamensresulatet bliver ændret til -3, og i særligt grove tilfælde kan eksaminanden blive bortvist fra universitetet. Dette er sket, så det er ikke kun en tom trussel.

2 Aflevering

Besvarelsen skal afleveres elektronisk via Absalon efter følgende regler.

2.1 Afleveringsfrist

Den *ordinære* afleveringsfrist er **fredag, den 9. november, kl. 9:00**. Hvis og *kun hvis* (!) du enten

- Deltager i en anden eksamen d. 7. eller 8. november 2012, eller

- Deltager i undervisning på MatIntro en af disse dage,

så er din afleveringsfrist fredag, den 9. november, kl. 23:55. Denne *ekstraordinære* afleveringsfrist finder kun anvendelse for MatIntro-studerende og studerende med anden eksamen d. 7–8. november for at sikre lige vilkår for alle eksamensdeltagere i deres respektive eksamensperioder. Hvis du er MatIntro-studerende er det *ikke* nødvendigt at søge om dispensation. Er du indskrevet på et andet for din uddannelse væsentligt *kursus* med *forpligtende* kursusaktiviteter i den ordinære IP-eksamensperiode fra 7. november, kl. 9:00, til 9. november, kl. 9:00, kan du tilsvarende søge om forlænget eksamenstid ved fremlæggelse af dokumentation af: indskrivning, de forpligtende aktiviteter og din deltagelse i dem. Dokumentation skal fremlægges til den kursusanvarlige (evt. på email `torbenm@diku.dk`) *inden* udløb af den *ordinære* eksamensfrist.

Det bemærkes, at førsteårsstuderende på Datalogi som udgangspunkt bruger den *ordinære* frist. Det er kun, hvis du deltager i undervisning eller eksamen i andre universistetskurser end IP og DiMS, at det kan komme pa tale at bruge den sene frist.

2.2 Procedure

Eksamenssættet uploades efter samme procedure som aflevering af de obligatoriske opgaver på kurset. På kursets Absalon-hjemmeside findes menupunktet “Eksamen”, hvorunder opgavepunkter med titlerne “Aflevering af eksamen (ordinær frist)” og “Aflevering af eksamen (ekstraordinær frist)” forefindes. Du skal anvende førstnævnte, hvis din afleveringsfrist er kl. 9:00; sidstnævnte, hvis den er kl. 23:55. I tilfælde af—og *kun* i tilfælde af—at Absalon ikke fungerer i *hele* timen inden din afleveringsfrist, udskydes afleveringsfristen med en time til kl. 10:00, hhv. 00:55. Skulle Absalon heller ikke fungere i denne periode, skal besvarelsen umiddelbart efter udløb af fristen sendes med email til `torbenm@diku.dk`. Det anbefales, at man ikke venter til sidste øjeblik med at uploade sin besvarelse. Man kan evt. løbende uploade delvise besvarelser og erstatte dem med nyere versioner efterhånden.

Det er den studerendes eget ansvar at gøre sig bekendt med, om Absalon fungerer på afleveringstidspunktet.[1] Der kan forekomme mild overbelastning, hvis mange forsøger at aflevere eksakt samtidigt — du opfordres derfor til at uploade din besvarelse i så god tid som muligt.

Hvis du er tilmeldt eksamen, men ikke kurset, er det op til dig selv i god tid at bede en af underviserne om at oprette dig som deltager på Absalonsiden for kurset.

2.3 Format

Alle opgaverne skal afleveres i én fil navngivet “efternavn.fornavne.sml”. Hedder man f.eks. “Jakob Grue Simonsen”, skal filen således navngives “Simonsen.JakobGrue.sml”. Bemærk, at man skal angive sit *fulde* navn, dog kan ikke-standard tegn i navnet erstattes med lignende tegn. For eksempel kan accenter udelades.

Bemærk, at det er muligt for studerende at uploade mere end én fil. Den *senest rettidigt afleverede* fil — og kun den — vil blive anset for den endelige eksamensbesvarelse.

Det er afgørende, at filens indhold er et korrekt SML-program, der kan køre uden fejl under Moscow ML 2.00, Moscow ML 2.01 eller Moscow ML 2.10 ved hjælp af kommandoen `mosml -P full`.

Det er tillige et krav, at funktioner i filen har *præcis de navne og typer*, der er specificeret i opgaveteksten, også hvad angår små og store bogstaver. I modsat fald kan man risikere, at hele eksamensbesvarelsen vil blive betragtet som ukorrekt. Hvis man har en delvis løsning til en

[1] Det er værd at bemærke, at Absalon indtil nu har fungeret upåklageligt i alle eksamensperioder.

delopgave, men at denne af en eller anden grund ikke kan køres uden fejl, skal denne indsættes i SML-kommentarer `(* ... *)`.

Brug af funktioner fra standardbiblioteket som beskrevet på `http://www.itu.dk/people/sestoft/mosmllib/index.html` er tilladt (og anbefales, hvor relevant). Bemærk, at modulnavne skal skrives med den samme kombination af store og små bogstaver som beskrivelsen i manualen. Da Windows ikke kender forskel på store og små bogstaver i modulnavnene, bør brugere af Moscow ML på Windows være særligt opmærksomme på dette, da der under Windows ikke vil rapporteres fejl, hvis man f.eks. skriver `list.concat` i stedet for `List.concat`. Men det vil det i opgaveretternes testsystem.

I opgaver, hvor man bliver bedt om at skrive tekst (for eksempel forklaringer eller visning af evalueringstrin), der ikke kan afvikles under Moscow ML, skal denne tekst tillige indsættes i SML-kommentarer.

Alle funktioner i besvarelsen forventes kommenteret i henhold til god kommentarskik, se evt. IP-2, Afsnit 5.3.3, og al programtekst skal opstilles pænt med passende indrykning, og hver linje må være maksimum 80 tegn lang inklusive indrykning, hvor et tabulatortegn tæller som 6 mellemrumstegn. (Det er dog bedst at undgå brugen af tabulatortegn). Stilkarakterer kan tælle op til 20% af den samlede besvarelse.

Kommentarer kan skrives på dansk eller engelsk efter eget valg.

Eksaminanden opfordres kraftigt til at afprøve sine funktioner for at sikre korrekte besvarelser. Medmindre det er eksplicit forlangt i opgaven, behøves afprøvningen dog ikke inkluderes i besvarelsen.

3 Støtte i eksamensperioden

Spørgsmål om eventuelle uklarheder i opgaveteksten samt spørgsmål om formalia i forbindelse med eksamen kan stilles til instruktorvagten i DIKUs kantine den 7. og 8. november i perioden 9:00-18:00 samt på kursets elektroniske diskussionsforum under Absalon.

Opdaterede versioner af opgavesættet med eventuelle rettelser og besvarelser af relevante spørgsmål til instruktorerne og på diskussionsforummet lægges på Absalon under menupunktet "Eksamen" på følgende tidspunkter:

- onsdag, 7. november, kl. 12:00
- onsdag, 7. november, kl. 16:00
- torsdag, 8. november, kl. 12:00

Disse og *kun* disse gælder som supplerende oplysninger om eksamenen. Det er den enkelte studerendes ansvar at holde sig ajour med disse opdateringer. Indlæg på diskussionsforummet er ikke autoritative og kan i værste fald være misvisende. Da væsentlige spørgsmål, rettelser og kommentarer fra diskussionsforummet medtages i opdateringerne, er det således ikke nødvendigt løbende at følge diskussionsforummet.

Kun spørgsmål og meddelelser med fortroligt indhold rettes direkte til den kursusansvarlige, Torben Mogensen, tlf. 21849672, e-mail `torbenm@diku.dk`, lokale 3-1-17 indenfor almindelig arbejdstid (9-17). Torben vil ikke være tilgængelig i hele denne periode, men vil besvare email så hurtigt som muligt.

4 Eksamensopgaver

Denne og de følgende syv sider indeholder de 8 eksamensopgaver, som skal løses, og hvis besvarelse skal afleveres i henhold til ovenstående instruktioner.

Selv om delopgaverne er af forskellig størrelse og sværhedsgrad, vægtes de nogenlunde ligeligt (4% – 5% per delopgave). Det er dog nemmere at få point for delvist korrekte besvarelser for de større delopgaver.

I sammentællingen tæller korrekthed ca. 80%, mens der gives ca. 20% i stilkarakterer. Se evt. den *SML style guide*, der kan findes på kursussiden.

Opgave 1 Et *palindrom* er en tekst, der læses ens forfra og bagfra, som for eksempel `"pip"` eller `"regninger mellem regninger"`. Både den tomme tekst `""` og tekster med et enkelt tegn (som f.eks. `"?"`) regnes som palindromer.

(a) Erklær i SML en funktion `erPalindrom : string -> bool`, sådan at `erPalindrom` *t* er `true`, hvis *t* er et palindrom og `false`, hvis *t* ikke er et palindrom.

(b) Et *udvidet palindrom* er en tekst, der er et palindrom, hvis man ser bort fra følgende forskelle:

- Store eller små bogstaver.
- Blanktegn.
- Tegnsætningssymboler (`#"."`, `#","`, `#"?"`, `#"!"`, `#":"` og `#";"`).

For eksempel er teksterne `"A Toyota"` og `"A man, a plan, a canal: Panama!"` udvidede palindromer.

Erklær i SML en funktion `erUdvidetPalindrom : string -> bool`, sådan at `erUdvidetPalindrom` *t* er `true`, hvis *t* er et udvidet palindrom og `false`, hvis *t* ikke er et udvidet palindrom.

Opgave 2 Vi definerer datatypen:

```
datatype rute = Stop | Frem of int * rute | Drej of int * rute
```

som beskriver en rute på et kort:

- `Stop` markerer enden på ruten.
- `Frem` (d, r) betyder, at man bevæger sig d meter fremad og derefter følger ruten r. d skal være ikke-negativ.
- `Drej` (g, r) betyder, at man drejer sig g grader mod uret, og derefter følger ruten r. Hvis g er negativ ($g = -h$), betyder det, at man drejer h grader med uret.

En rute siges at være *normaliseret*, hvis der *ikke* er delruter af en af formerne

1. `Frem (0,` r`)`
2. `Drej (0,` r`)`
3. `Drej` (g, r), hvor $g \leq -180$ eller $g > 180$
4. `Frem` (d_1, `Frem` (d_2, r))
5. `Drej` (g_1, `Drej` (g_2, r))

Da disse ville kunne forkortes til henholdsvis

1. r
2. r
3. `Drej` (g', r), hvor $-180 < g' \leq 180$, og $g - g'$ er deleligt med 360.
4. `Frem` ($d_1 + d_2$, r)
5. `Drej` ($g_1 + g_2$, r)

(a) Erklær i SML en funktion `korrekt : rute -> bool`, sådan at `korrekt` r er `false`, hvis r indeholder en delrute af formen `Frem` (d, r), hvor $d < 0$, og `true`, hvis dette ikke er tilfældet.

(b) Erklær i SML en funktion `laengde : rute -> int`, sådan at `laengde` r er den samlede afstand, ruten r tilbagelægger, dvs. summen af alle d_i, hvor `Frem` (d_i, $\cdots$) forekommer i r. Du kan antage, at `korrekt` r = `true`.

(c) Erklær i SML en funktion `erNormaliseret : rute -> bool`, sådan at `erNormaliseret` r er `true`, hvis r er en normaliseret rute, og `false` ellers. Du kan antage, at `korrekt` r = `true`.

(d) Erklær i SML en funktion `normaliserRute : rute -> rute`, der bruger ovenstående forkortelsesregler til at normalisere en rute. Det skal altså gælde, at `erNormaliseret (normaliserRute` r`) = true` for alle ruter r. Endvidere skal `normaliserRute` r repræsentere den samme rute i den forstand, at en person, der følger ruten vil gå den samme vej (men måske ikke snurre helt så meget rundt i svingene). Kort sagt, skal de ovenstående forkortelsesregler og ingen andre bruges til at forkorte ruten til en normaliseret rute. Du kan antage, at `korrekt` r = `true`.

Opgave 3 Et heltal $n > 0$ siges at være *kvadratfrit*, såfremt intet heltal af formen m^2, hvor $m > 1$, er divisor i n. For eksempel er 21 kvadratfri, da $21 = 3 \times 7$, mens 54 ikke er kvadratfrit, da $54 = 2 \times 3^3$ og derfor har 3^2 som divisor.

(a) Erklær en funktion `kvadratfrit : int -> bool` i SML, således at `kvadratfrit` n returnerer `true` hvis $n > 0$ og n er kvadratfrit. Hvis $n > 0$ og n ikke er kvadratfrit, returneres `false`. Hvis $n \leq 0$, skal `kvadratfrit` n kaste undtagelsen `Domain`.

(b) Vis beregningstrinnene for de to udtryk `kvadratfrit 21` og `kvadratfrit 54`. Du skal blot vise sekvensen af funktionskald og returværdier – du behøver ikke at vise beregningerne mellem funktionskald. Hvis du bruger biblioteksfunktioner, skal du ikke vise de kald, som biblioteksfunktionerne laver til andre funktioner.

(c) Erklær en funktion `maksKvadratfrit : int -> int` i SML, således at `maksKvadratfrit` n returnerer det største kvadratfri tal k, der er divisor i n. For eksempel skal `maksKvadratfrit 21` returnere 21, da 21 er kvadratfri (og divisor i 21), mens `maksKvadratfrit 54` skal returnere 6, da 6 er den største kvadratfri divisor i 54. Hvis $n \leq 0$, skal `maksKvadratfrit` n kaste undtagelsen `Domain`.

I første og tredje delopgave bør køretiden ikke overstige et sekund for tal under ti cifre (som f.eks. 999999937). Vink: Hvis x^2 går op i n er $x \leq \sqrt{n}$.

Opgave 4 En liste P er en *permutation* af en liste L, såfremt P kan laves ud fra L ved at bytte om på rækkefølgen af elementer i L. For eksempel er listen `[3, 7, 4, 7]` en permutation af listen `[7, 7, 3, 4]`. Mere præcist er P en permutation a L, hvis begge følgende ting gælder:

1. Hvis et listeelement x forekommer n gange i L, forekommer x også n gange i P.

2. Hvis et listeelement x forekommer n gange i P, forekommer x også n gange i L.

(a) Erklær en funktion `erPermutationAf : ''a list * ''a list -> bool` i SML, således at `erPermutationAf (`P`,`L`)` returnerer `true` hvis P er en permutation af L.

(b) Erklær en funktion `antalPermutationer : ''a list -> int` i SML, således at `antalPermutationer` L returnerer antallet af *forskellige* permutationer af L. `antalPermutationer` L kan kaste undtagelsen `Overflow`, hvis der under beregning forekommer tal, der ikke kan repræsenteres som heltal i Moscow ML.

Bemærk, at antallet af permutationer af L ikke kun er en funktion af antallet af elementer i L, da gentagne elementer nedsætter antallet af mulige permutationer. For eksempel har listen `[42,42,42,42]` kun en permutation (listen selv), og listen `[true,true,false,false]` har seks permutationer:

```
[true,true,false,false]
[true,false,true,false]
[true,false,false,true]
[false,true,false,true]
[false,true,true,false]
[false,false,true,true]
```

Vi erindrer om, at hvis alle elementerne i en liste L af n elementer er forskellige, findes der $n!$ (n fakultet) forskellige permutationer af L. Da både `true` og `false` forekommer to gange i listen `[true,true,false,false]` (som har længde 4), er det samlede antal forskelige permutationer $4!/(2! * 2!) = 6$. Generelt vil en liste, der har i alt n elementer bestående af m forskellige elementer $x_1, \ldots, x_m$, hvor hvert element x_i forekommer k_i gange, have $n!/(k_1! \cdot \cdots \cdot k_m!)$ forskellige permutationer. F.eks. vil antallet af permutationer af listen `["a", "a", "a", "b", "b", "c", "d", "d"]` være $8!/(3! * 2! * 1! * 2!)$.

(c) Antallet af permutationer af listen

`[2, 2, 2, 2, 2, 2, 2, 2, 3, 5, 5, 5, 7, 7, 7, 7, 7, 7, 7, 7, 7]`

er $21!/(8! * 1! * 3! * 9!) = 581981400$, men en naiv beregning af dette tal vil kaste undtagelsen `Overflow`, da 21! er for stort til at repræsenteres som et heltal i Moscow ML. Men det samlede resultat 581981400 kan godt repræsenteres som et heltal i Moscow ML. Denne opgave går derfor ud på at lave beregningen, så overløb undgås, med mindre det endelige resultat er for stort til at repræsenteres som et heltal i Moscow ML.

Erklær en funktion `antalPermutationerNy : ''a list -> int` i SML, således at `antalPermutationerNy` L returnerer antallet af *forskellige* permutationer af L. Hvis antallet af permutationer i L overstiger det største heltal i SML, skal `antalPermutationerNy` L kaste undtagelsen `Overflow`, men ellers skal der ikke rejses undtagelser. Det er *ikke* tilladt at bruge kommatal (`real`) i delberegningerne.

Vink: Forkort brøken, inden faktorerne i tæller og nævner ganges sammen. Brug evt. en funktion `forkort : int list * int -> int list`, hvor `forkort ([`$t_1, \ldots, t_n$`],`n`) = [`$t'_1, \ldots, t'_n$`])`, sådan at $t'_1 \cdot \ldots \cdot t'_n = (t_1 \cdot \ldots \cdot t_n)/n$. Du kan med fordel bruge funktion `gcd` fra side 18 i Hansen & Rischel.

Opgave 5 Denne opgave omhandler *polymorfe højereordensfunktioner*, så din løsning bør vise, at du forstår at bruge og definere sådanne.

(a) Erklær i SML en funktion `grupper : ('a -> ''b) -> 'a list -> 'a list list`, således at kaldet `grupper` f l returnerer en liste af lister ll, som tilsammen har præcis de samme elementer som l i samme antal[2], men sådan at hver liste i ll består af elementer fra l, der giver samme værdi, når f anvendes på dem, og ydermere sådan, at alle elementer fra l, der giver samme værdi, når f anvendes på dem, ligger i samme liste i ll. Men andre ord er antallet af lister i ll minimalt.

For eksempel kan `grupper Int.sign [0,3,~2,5,0,1,~7]` give resultatet `[[0,0],[3,5,1],[~2,~7]]`, idet der grupperes efter værdien af `Int.sign` anvendt på elementerne. Bemærk, at der ikke er krav til rækkefølgen af dellister eller til rækkefølgen af elementer i dellisterne, så et lige så godt resultat kunne være `[[5,1,3],[~7,~2],[0,0]]`.

(b) Erklær i SML en funktion `gentag : ('a -> 'a) -> 'a -> 'a`, sådan at `gentag` f x anvender f gentagne gange på x indtil dette giver en undtagelse. Det sidste argument før undtagelsen returneres.

Med andre ord beregnes sekvensen $x_0 = x,\ x_1 = f(x_0), \ldots,\ x_{i+1} = f(x_i), \ldots$, og hvis $f(x_j)$ kaster en undtagelse (og ingen tidligere anvendelse gjorde det), returneres x_j.

For eksempel vil `(#2) o gentag (fn (x::xs, ys) => (xs, x::ys))` være ækvivalent med funktionen `itrev` på side 256 i Hansen & Rischel, idet undtagelsen `Match` vil rejses, når det først argument er den tomme liste.

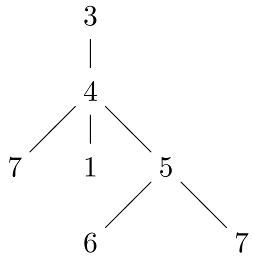
(c) Definer ved brug af `gentag` (og uden eksplicit rekursion) en funktion `gcd : int * int -> int` ækvivalent til funktionen med samme navn på side 18 i Hansen & Rischel.

[2] Med andre ord er `List.concat` ll en permutation af l.

Opgave 6 Generelle træer kan have værdier i alle knuder i træet, og hver knude kan have vilkårligt mange børn. Vi kan definere en datastruktur for sådanne træer:

```
datatype 'a trae = K of 'a * ('a trae list)
```

(a) Skriv en SML-erklæring af en variabel `t7 : int trae`, der repræsenterer træet herunder:

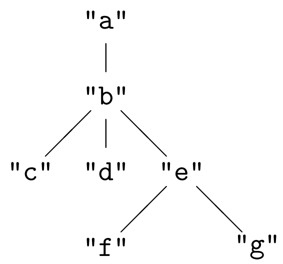


(b) Et *præordensgennemløb* (*preorder traversal*) af et træ, er et gennemløb af knuderne i et træ, så knuden besøges før sine børn, som besøges i rækkefølge fra venstre mod højre, idet hvert barn også gennemløbes med præordensgennemløb.

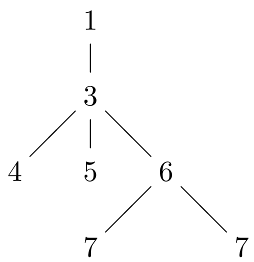
Erklær i SML en funktion `praeorden : 'a trae -> 'a list`, sådan at `praeorden` t er en liste af knudeværdierne i t i den rækkefølge, der vil blive besøgt i et præordens gennemløb af t. For eksempel skal `praeorden t7` give listen `[3, 4, 7, 1, 5, 6, 7]`.

(c) Erklær i SML en funktion `erstat : 'a trae * 'b list -> 'b trae * 'b list`, sådan at `erstat (`t`, `l`)` er et par `(`t'`, `l'`)`, hvor t' er et træ af samme struktur som t, men hvor knudernes værdier er erstattet med værdier fra l, sådan at `(praeorden `t'`) @ `$l' = l$. Hvis længden af l er mindre end længden af `praeorden` t, skal der kastes en undtagelse. Det er ikke vigtigt hvilken.

For eksempel skal kaldet `erstat t7 ["a","b","c","d","e","f","g","h"]` give parret `(`$t7'$`,["h"])`, hvor $t7'$ er træet



(d) Erklær i SML en funktion `sorter : int trae -> int trae`, sådan at `sorter` t er et træ med samme struktur som t, som indeholder de samme tal i samme antal som t, men hvor tallene er anbragt sådan, at `praeorden (sorter `t`)` er en sorteret liste. For eksempel skal `sorter t7` give følgende træ:



Opgave 7 Et meget simpelt format for billedfiler er PPM („Portable Pixmap"), som er beskrevet blandt andet på `http://paulbourke.net/dataformats/ppm/`.

En PPM-fil er en tekstfil, der består af følgende elementer i rækkefølge:

1. Teksten `P6`.
2. Et heltal (som tekst) w, som er bredden af billedet målt i antal pixels.
3. Et heltal (som tekst) h, som er højden af billedet målt i antal pixels.
4. Teksten `255`
5. $3 \cdot w \cdot h$ bytes (elementer af SML-typen `char`), hvor hver gruppe af tre bytes r, g, b beskriver en pixel som intensiteten af de tre grundfarver rød, grøn og blå som værdier mellem 0 og 255. For eksempel vil en pixel med intensiteterne 57, 40 og 191 i henholdsvis rød, grøn og blå blive repræsenteret med teksten `9(¿`, da tegnene i denne tekst netop har koderne 57, 40 og 191 ved brug af ISO 8859-1 indkodningen (se s. 20 i IP-2).

De fire første elementer kan adskilles af blanktegn, linjeskift eller en kombination af disse. Mellem disse elementer kan indsættes kommentarer, der starter med et `#` og slutter ved næste linjeskift. Efter det fjerde element (`255`) skal der være *præcis* et tegn (typisk et blanktegn eller et linjeskift) inden pixelværdierne. **I denne opgave kan du antage, at tegnfølgen `255` ikke forekommer før i element 4, så første pixelværdi ligger fire bytes efter starten af den første forekomst af tegnfølgen `255`.** Et eksempel på en simpel PPM-fil er vist herunder:

```
P6
# Simpelt billede
3 2 # 3x2 = 6 pixels
255
9(¿9(¿9(¿9(¿9(¿9(¿
```

Bemærk, at teksten `"9(¿9(¿9(¿9(¿9(¿9(¿"` indeholder præcis 18 bytes, som tilsammen beskriver 6 ens pixels med intensiteterne 57, 40 og 191 for rød, grøn og blå. Eksemplet her bruger kun „skrivbare" tegn, men alle 256 tegn/koder fra ISO 8859-1 kan forekomme i PPM-filer. To eksempelfiler `rgb.ppm` og `kort.ppm` findes sammen med opgaveteksten.

(a) Erklær i SML en funktion `inverterPPM : string -> string -> unit`, sådan at `inverterPPM` *fi fo* læser et PPM-billede fra filen *fi* og skriver et PPM-billede til filen *fo*. Billedet i *fo* skal have alle pixels inverteret i forhold til *fi*, dvs. at hver grundfarveværdi c skal erstattes af værdien $255 - c$. Hvis det ikke lykkes at åbne filerne eller hvis der er fejl i filformatet for *fi*, er det udefineret, hvad der sker.

Hvis ovenstående eksempel findes i filen `simpel.ppm`, skal kaldet `inverterPPM "simpel.ppm" "lepmis.ppm"` danne en ny fil `lepmis.ppm`, som har indholdet

```
P6
# Simpelt billede
3 2 # 3x2 = 6 pixels
255
Æ×@Æ×@Æ×@Æ×@Æ×@Æ×@
```

Bemærk, at teksten til og med tegnet efter `255` er kopieret uændret, mens alle efterfølgende tegn er inverterede. På grund af antagelsen beskrevet før, vil dette altid gælde.

Opgave 8 Vi har givet denne signatur for *symboltabeller*:

```
signature SYMBOLTABEL =
sig
  type 'vaerdi tabel
  val tom : 'vaerdi tabel
  val indsaet : 'vaerdi tabel -> (string * 'vaerdi)
                  -> 'vaerdi tabel
  val find : 'vaerdi tabel -> string -> 'vaerdi option
end
```

En symboltabel binder nøgler (af typen `string`) til værdier (af typen `'vaerdi`). Operationen `indsaet` indsætter en binding (angivet som et nøgle/værdi-par (n, v)) i tabellen og `find` finder den til en nøgle n bundne værdi v. Denne er givet som `SOME` v, hvis n er bundet til værdien v, og som `NONE`, hvis n ikke er bundet til nogen værdi. Følgende skal gælde:

- `find tom` n = `NONE`
- `find (indsaet` t `(`n`,` v`))` n = `SOME` v
- `find (indsaet` t `(`n'`,` v`))` n = `find` t n, hvis $n \neq n'$.

(a) Skriv en struktur `ListeTabel`, der implementerer signaturen `SYMBOLTABEL` ved at bruge en liste af nøgle/værdi-par.

(b) Skriv en struktur `FunTabel`, der implementerer signaturen `SYMBOLTABEL`, sådan at en `tabel` er en funktionsværdi. Dvs, at strukturen indeholder linjerne

```
type 'vaerdi tabel = string -> 'vaerdi option
fun find t = t
```

(Opgavesættet slut)

5 Rettelser og klarifikationer til det oprindelige sæt

- I opgave 8 passede reglerne for `find` ikke med siganturen, idet parametrene var byttet om. Det er nu rettet.
- Alle opgaver kan løses uden, at man løber tør for lager, hvis man har min. 2GB lager på sin maskine. Men hvis man i opgave 7 kan klare de to små billeder uden fejl men ikke *kan* undgå at løbe tør for lager med det store billede, så vil det kun trække *lidt* ned.