**Spørgsmål til reeksamen i Programmering og Problemløsning.**

* Funktionsprogrammering: Typer, tupler, records, og tagged values (mærkater)
* Funktionsprogrammering: Lister, mængder, afbildninger (maps)
* Funktionsprogrammering: Træer (finite trees)
* Imperative programmering: Variable, sekventiel eksekvering og arrays
* Objektorienteret programmering: Klasser og objekter
* Objektorienteret programmering: Nedarvning

**Funktionsprogrammering: Funktioner, Simple og sammensatte typer:**

Opgave:

Giv en kort beskrivelse af begreberne funktion og type, og forklar hvad forskellen mellem simple typer og sammensatte typer (compound types). Giv eksempler på funktionsdefinitioner og forklar hvad rekursion er. Giv eksempler på alle ovenstående emner og hvordan de kan bruges sammen.

Lærervejledning:

Det forventes at der bliver talt om emner fra Kap. 2 og 3. Minimum er diskussion af funktionsdefinitioner (anonym, function-keyword og match-with) inkl. mønstergenkendelse (pattern matching) og tilhørende guards og rec nøgleordet. Som minimum skal der diskuteres typerne int, float, tupler, records og tagged values (mærkater). Avancerede emner er: 1) overloading af funktioner og operatorer; 2) sammensætning af funktioner inkl. præcedens og piping; 3) funktioner med tuple argument versus flere argumenter og currying; 4) sammenligningsoperatorer og ordning; 5) enumererede typer; 6) polymorfi; 7) fejlhåndtering med ”failwith”.

**Funktionsprogrammering: Lister, mængder, afbildninger (maps):**

Opgave:

Beskriv datatyperne list, endelige mængder (finite sets) og afbildninger (maps), og beskrive deres tilhørende hyppigt anvendte operatorer. Sammenlign deres ligheder og forskelle ud fra.

Lærervejledning:

Det forventes at der bliver talt om emner fra Kap. 4 og 5. Minimum er diskussion af lister (inkl. list nøgleord, cons-operator, list-mønstre, konkatenationsoperator, rekursion over listestrukturer), sæt (inkl. set nøgleord, forskellen mellem lister og sæt) og afbildninger (inkl. map nøgleord, forskellen mellem lister og afbildninger). Avancerede emner er 1) lists: tegning af cons operator som et træ, sammenligning, polymorfi, List.map, fold, foldBack; 2) hvordan man kan implementere sæt og afbildninger med lists.

**Funktionsprogrammering: Træer (finite trees):**

Opgave:

Beskriv datatypen træer, giv eksempler på hvordan træer defineres og benyttes, hvordan funktioner der tager træer som argumenter erklæres. Giv eksempler på, hvordan binære træer kan gennemløbes og diskuter deres forskelle.

Lærervejledning:

Det forventes at der bliver talt om emner fra Kap. 6. Minimum er diskussion af rekursiv type-erklæring, mønstergenkendelse, funktionserklæring, pre-, in- og post-order gennemløb (tree traversals). Advancerede emner er: 1) polymofisk type-erklæring, træer med variabelt antal sub-træer

**Imperative programmering: Variable, sekventiel eksekvering og arrays:**

Opgave:

Beskriv forskellen mellem værdier og variable. Beskriv hvordan imperativ programmering kan gennemløbe en liste uden brug af rekursion. Diskuter forskellen mellem funktions- og imperativ programmeringsparadigmet.

Lærervejledning:

Det forventes at der bliver talt om emner fra Kap. 8. Minimum er diskussion af variable (inkl. mutable nøgleordet), sideeffekter, ”<-” operatoren, sekventiel sammensætning af udtryk med ”;” operatoren, while-løkker, tabeller (arrays). Avancerede emner er: 1) mutable records (og de tilsvarende references ”ref” operatorer); 2) forskellen mellem rekursiv og imperativ gennemløb af lister

**Objektorienteret programmering: Objektorienteret modellering:**

Opgave:

Beskriv hvordan man strukturerer en løsning af et problem ved brug af den objektorienteret designprocess. Giv et eksempel og diskuter eventuelle fordele og ulemper ved denne metode.

Lærervejledning:

Det forventes at der bliver talt om emner fra ”Modeling the Static/Data Aspects of the System”. Minimum er diskussion af navne-ord-som-klasser-og-udsagnsord-som-metoder metoden, UML diagrammer, eksempel på en simpel klasse, konstruktor-begrebet, brugen af member nøgleordet, fordelen og ulemper ved brug af oopd til udvikling af store programmer. Avancerede emner er: 1) forskellen på let og val nøgleord; 2) forskellen på implicite og eksplicite konstruktorer; 3) nedarvning; 4) vedligeholdelse og test.

**Objektorienteret programmering: Nedarvning:**

Opgave:

Beskriv begrebet nedarvning i den objektorienteret programmerings og designmetode, og diskuter fordele og ulemper ved at strukturere et program der gør brug af nedarvning.

Lærervejledning:

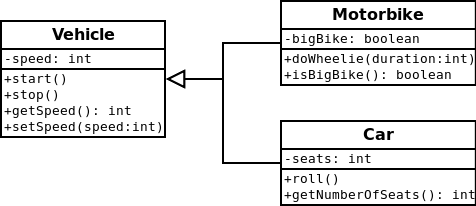
Det forventes at der bliver talt om emner fra ”Modeling the Static/Data Aspects of the System” og fsharpforfunandprofit.com/Inheritance and abstract classes. Minimum er diskussion af eksempler der gør brug af nedarvning, UML diagrammer, fordele og ulemper for debuggingprocessen. Avancerede emner er: 1) abstract og AbstractClass nøgleorderne, 2) overskrivning af nedarvede værdier og metoder;

Use-cases og programmeringsopgaver til evt. brug under eksamen.

* Programmeringsopgave:  
  Lave et program som udskriver tallene fra 0 til 10 på skærmen med brug af rekursion og derefter med brug af en while-løkke.
* Use-case: Stack  
  A stack is a “last in, first out” linear data structure. A stack can have any object as an element. It has two fundamental operations, called *push* and *pop*. The *push* operation adds a new item to the top of the stack. If the space allocated to hold the stack is full when the push operation is attempted, then an error condition is raised. The *pop* operation removes an item from the top of the stack. A *pop* reveals previously concealed items, or results in an empty stack. If the stack is empty when a *pop* operation is attempted, then an error condition is raised.

Implement a stack using OOP. Your code should store the stack elements in an array and should not make use of a stack class from a class library.

* Programmeringsopgave  
  Implement the following classes and add an abstract class of your own choice.



* Use-case: Online auktion  
  En online auktion er et offentligt salg af f.eks. ting eller tjenesteydelser til højstbydende på nettet. Auktionen forestås auktionarius, som er en betroet mellemmand eller system. Auktionarius modtager varen til salg fra sælger, udstiller varen, modtager bud fra køber, og afgør vinderen.   
    
  En typisk auktion vil bestå af en udstillingsperiode for et antal vare, og dernæst et købsvindue, hvor købere kan byde på en bestemt vare. Køberne kan se hinandens bud, og den, som indenfor købsvinduet har budt højest, har købt produktet. Køber betaler det budte beløb til sælger, og et auktionssalær til auktionarius samt moms til staten.

* Use-case: Spillet dam  
  Spilles af 2 spillere, der hver får 12 brikker, som anbringes i de 3 første rækker på de mørke felter. Der spilles kun på de mørke felter, og brikken flyttes skråt fremad 1 felt ad gangen.   
    
  Modstanderens brikker slås, hvis disse støder op til ens egne med et tomt felt skråt bagved. Flere brikker kan slås i samme træk, bare der hele tiden er et tomt felt bag den brik, som erobres. Man kan skifte retning for hver brik man har slået, så længe man bevæger sig fremad.   
    
  Lykkes det at komme frem til modstanderens bageste række, opnår man en dam. Denne brik markeres ved at placere en af ens slagne brikker ovenpå. En dam kan flyttes på skrå lige så mange felter frem eller tilbage, som der er tomme felter, og slå modstanderens brikker som tidligere.   
    
  Spilleren, der har slået alle modstanderens brikker, eller har lukket dem inde, så de ikke kan flyttes, har vundet
* Use-case: Kalender  
  En kalender er et værktøj til at organisere dage for sociale, administrative eller andre opgaver. Kalender består af dage organiserede i uger, måneder og år. Til hver dag er der tilknyttet en unik dato, og hver dag neddeles yderligere i timer. Det er muligt at opskrive aftaler i kalenderen, og derudover har nogle dage særlige mærker såsom helligdage, månens cyklus el. lign.. Kalendere kan have fælles aftaler, og disse aftaler kan tilknyttes særlig information, såsom mødested og mødemateriale. En person kan have flere kalendere, og en kalender kan deles mellem flere personer.
* Use-case: kortspillet krig  
  Hver spiller får 26 kort, der placeres foran spilleren i én bunke med billedsiden nedad. Spillerne vender nu begge det øverste kort. Den, der har det højeste kort, vinder modstanderens kort.  
    
  Når spillerne vender kort af samme værdi, bliver der krig. Begge lægger nu yderligere 3 kort frem og der tredje afgør, hvem der vinder krigen. Vinderen tager alle de vendte kort fra modstanderen. Hvis det tredje kort ved hver spiller har samme værdi igen, fortsætter krigen og der vendes yderligere 3 kort (6 i alt pr. spiller).   
    
  Vundne kort placeres med bagsiden opad i bunden af vinderens kortstak. Der fortsættes til én af spillerne har vundet alle modstanderens kort.
* Use-case: Sænke slagskibe  
  Dette er et spil for to personer, der kan spilles med papir og blyant. Der spilles på fire plader, to for hver spiller, og hver plade er inddelt i 10x10 felter. Hvert felt identificeres vha. dets række og søjle nummer.  
    
  Hver spiller for tildelt et antal skibe, som placeres på spillerens ene plade og markerer, hvor modstanderen har forsøgt at skyde. På den anden plade markerer spilleren tilsvarende, hvor han/hun har forsøgt at ramme modstanderen.  
    
  Når skibene er placeret skiftes spillerne til at skyde på modstanderens felt, og modstanderen annoncerer ramt eller plask, alt efter om et skib blev ramt eller ej. Vinderen er den, der først får sænket alle modstanderes skibe.