Den ukendte eksamensopgave i programmering B

Søren Præstegaard, spr@sde.dk

Forår 2019

Til eksamen i programmering på B-niveau skal eleverne bruge 60 minutters forberedelsestid på at arbejde med en ukendt opgave. På FIP-kurserne i foråret 2019 afholdtes en workshop, der omhandlede udformningen af disse opgaver. Deltagerne formulerede et antal bud på eksamensopgaver, og det er de opgaver du kan læse i dette dokument. Opgaverne skal tjene som eksempler, der kan bruges til at formulere opgaver der passer til ens egen eksamen, og da der givetvis mange måder man kan udvikle disse eksamensopgaver, står vi som fagkollegaer overfor en opgave, i at udforske mulighederne og skabe konsensus. Disse eksempler repræsenterer et fælles udgangspunkt, vel vidende at ingen af os på nuværende tidspunkt har prøvet denne eksamensform i praksis.

Diskussionspunkter

Deltagerne på workshoppen diskuterede også de muligheder og udfordringer der kan være ved disse opgaver. I punktform herunder følger de væsentligste pointer:

- Det er vigtigt at opgaverne indeholder konkrete faglige begreber. Selv blandt underviserne var der ofte stor forskel på tolkningen af enkelte delspørgsmål, hvis ikke der stod helt præcist hvad der forventes af eleven. Vi må formode at der blandt eleverne er endnu bredere rammer for denne fortolkning, og ord som *flowdiagram*, *skitse*, *funktion*, osv., hjælper især de fagligt svage elever med at arbejde relevant med opgaverne.

- Tydeligheden gælder også instruksen til eleven: Hvornår skal der programmeres? Hvornår skal der analyseres? Eksamensopgaverne skal hjælpe med at afdække elevernes opfyldelse af de faglige mål, og hvis man ønsker at eleverne arbejder med en bestemt programmeringsaktivitet bør man fortælle det tydeligt. Ellers risikerer man at opgaven ikke opfylder det formål man havde i tankerne.

- Man bør gøre sig det klart – og forventningsafklare med både elever og censor – i hvor høj grad man forventer at eleverne arbejder med alle

delspørgsmål i den stillede opgave. Dette bør også kommunikeres tydeligt til eleverne, både i formuleringen af opgaverne og som forberedelse til eksamen.

- Det er en udfordring af stille opgaver der er relevante for både de stærkeste og de svageste elever, men da opgaverne tildeles ved lodtrækning, bør man forsøge at sikre sig, at alle eleverne kan besvare opgaven i et eller andet omfang. Mange af de udviklede eksempler har et delspørgsmål med en højere grad af åbenhed, netop for at imødekomme dette.

- I skrivende stund er reglerne for eksamen sådan, at eleverne har adgang til internettet i forberedelseslokalet. De må naturligvis ikke kommunikere med andre, men de har mulighed for at søge dokumentation, eksempler, osv. Man bør forberede eleverne på hvordan de bruger denne mulighed, og hvad de skal passe på.

- Opgaverne bør afspejle at programmeringsfaget dækker flere forskellige programmeringshandlinger, og ikke blot *at skrive kode*. Det er ligeså relevant at læse kode, rette i kode, producere abstrakt dokumentation, osv.

- Indholdet i programmeringsfaget er meget forskelligt på de enkelte skoler, og spørgsmålene har derfor meget forskellig karakter. Det er vigtigt at de opgaver man stiller sine elever, har tæt forbindelse til de undervisningsforløb man har lavet i løbet af de to år.

Eksempel 1

Et fysikeksperiment har resulteret i en mængde målepunkter, og fysiklærerne ønsker nu at kunne behandle disse data på forskellige måder i et program. Det vedhæftede program åbner datafilen fra eksperimentet, laver en lille smule databehandlig og udskriver information i konsollen.

- I Undersøg koden og tegn et flowdiagram over programmets behandling af data fra filen.
- I Skriv en funktion, der kan tage en liste af målepunkter som argument og producere en graf over målepunkterne. Brug biblioteket `matplotlib` til at tegne grafen.
- I Udvid programmet med funktioner som du synes er passende for et sæt målepunkter fra fysiktimerne.

Til denne opgave hører et lille pythonprogram og en datafil. Der er naturligvis blevet arbejdet med python og matplotlib i undervisningen, ligesom eleverne har prøvet af tegne et flowdiagram, baseret på noget kode.

De faglige mål om samspil med andre fag og specifikationsmodeller kommer tydeligt i spil, og det åbne spørgsmål til sidst skal sikre at de dygtige elever kan vise hvad de kan.

Eksempel 2

Case: Skak

- Kør det udleverede program. Redegør for metoden `gyldigt_træk` i klassen Tårn, herunder eksempelvis variable, datatyper og forgreninger.
- Forklar begrebet nedarvning med udgangspunkt i superklassen Brik og underklassen Tårn.
- Skriv koden for metoden `gyldigt_træk` i klassen Bonde eller Springer.
- Beskriv mindst én måde et program som Skak kan testes på.

Gruppen har lagt vægt på, at progressionen i opgaven tilbyder eleven at komme godt i gang, hvor det er muligt at køre et udleveret program og redegøre for udleveret kode.

Krav til karakteren 12:

- Redegøre for både variable, datatyper og forgrening
- Bruge de rigtige begreber
- Plads til en lille fejl
- Én måde beskrives fyldestgørende

Krav til karakteren 02:

- Ikke perfekte besvarelser af de første to opgaver
- Kan redegøre nogenlunde for koden, hvis det ikke virker
- Kan ikke nødvendigvis redegøre en måde at teste

Eksempel 3

Cæsaralgoritmen
Cæsaralgoritmen flytter hvert bogstav et antal pladser ned i alfabetet. Eksemplet viser et skift på tre pladser, således at et B i klarteksten bliver til et E i cifferteksten.

Cæsaralgoritmen er et specialtilfælde af monoalfabetisk substitution, som blev benyttet under Julius Cæsar. Hvert bogstav erstattes med et bogstav et antal pladser længere nede i alfabetet (shift cipher). Nøglen er antallet af pladser der flyttes.

Et shift cipher for det danske alfabet er givet ved

$$e(x) = x + K \mod 29$$
$$d(x) = x - K \mod 29$$

F.eks. i teksten "cæsar"med K=3 bliver "c"til "F", "æ"til "A"osv., hvilket giver teksten: "FAVDU". Cæsaralgoritmen brydes relativt let ved frekvensanalyse.

- Tegn et flowdiagram over et program som kan kryptere beskeden: *Johnny bruger ikke gummistøvler* ved hjælp af caesar-kryptering.
- Udarbejd (**eller ret hvis underviseren vil stille et udgangspunkt til rådighed**) koden til programmet.
- Mere avanceret kryptering kan ske ved at man skifter hvert bogstav i en sætning eller streng individuelt. Tegn et flowchart for et program der kan håndtere skift af en tekststreng på et individuelt basis. Hvis man for eksempel har kodeordet hest, ville det blive til et skift på 8-5-19-20, der gentages igennem hele strengen.

(Den indledende tekst er klippet fra wikipedia: https://da.wikipedia.org/wiki/Krypteringsalgoritme, og er ikke nødvendigvis brugbar til en elevopgave. Eksemplet skal vise at opgaven starter med en beskrivelse af caesarkryptering.)

Eksempel 4

”Dagens bedste toilet!”
Når tiden kommer til at man skal, så skal man, men man vil gerne have den mest behagelige oplevelse. Kathrine vil gerne have det ”bedste toilet” og bruger gerne lidt ekstra tid på at finde det. Kan vi hjælpe hende med at udvikle et system som gør denne opgave nemmere?

- Hvilket krav/parametre skal et system som dette indeholde?
- Lav et flowchart som viser systemet.
- Hvilke datatyper er nødvendige?
- Programmer din løsning til Kathrines problem.
- Er der særlige hensyn vi skal tage højde for til brugergrænsefladen for dette system?/Tænk over eventuelle forbedringer til systemet.

Eksempel 5

Skriv et Blackjack-spil program eventuelt baseret på en klasse som kunne kaldes Kort.

Klassen har som minimum en privat variabel der betyder udtrykket hvilket kort der er tale om, en konstruktor som afsætter et tal fra 1 til 52 til variablen, og en træk funktion.

Målet i blackjack er at slå dealeren. For at gøre det skal du have en hånd, der scorer højere end dealerens hånd, men som ikke overstiger 21 i samlet værdi. Alternativt kan du vinde, når du har en hånd, der er mindre end 22, og dealerens hånd overstiger 21. Når den samlede værdi af din hånd er 22 eller højere, er du gået *bust*, og du taber automatisk dine satsede penge.

I hver blackjack runde spiller en spiler mod en dealer. Spilleren begynder med at lægge en indsats. Efter indsatsen er gjort, modtager spiller og dealeren hver ét kort, som er synlige for begge. Spillet venter på spiller: enten vælger han at få flere kort for at forbedre sin hånds samlede værdi, eller stopper. Når spilleren er færdig med at trække kort, fuldfører dealeren sin hånd.

- Tegn et flowchat diagram om spillet.
- Argumenter for nogle datatyper og prototyper af funktioner som man kan bruges i programmet. Du må komme ind på begreber som ”static” eller non - ”static” variabler og funktioner.
- Gør færdig en funktion der genererer et tilfældigt kort ad gangen ud af kort sættet, både for spiller og for dealer.
- Gør spillets overordnede logik færdig.

Eksempel 6

Lixtalsberegner.
Lixtal beregnes på følgende måde:

$$\text{Lixtal} = \frac{\text{antal ord}}{\text{antal sætninger}} + \frac{\text{antal lange ord} \cdot 100}{\text{antal ord}}$$

Lange ord defineres som ord med flere end seks bogstaver.

Når man lærer at læse i folkeskolen inddeles niveauerne i lixtal, hvilket betyder at bøger med mange korte ord har et lille lixtal. Det vil sige, at lixtallet fortæller noget om sværhedsgraden af teksten.

Der er lavet et stykke kode, der henter tekstfil ind og tæller antal af ord og punktummer. Der skal udvikles en funktion, som kan beregne lix. Du skal berøre alle fire spørgsmål, men gå i dybden med mindst ét.

- Lav et flowdiagram, der viser hvordan udregning af lixtal.
- Lav en pseudokode til programmet.
- Udvid programmet med en funktion, der kan tælle antallet af ord med flere end seks bogstaver.
- Udvid programmet, så det kan beregne lixtal.

Eksempel 7

Kaffeautomat
En kaffeautomat kan levere kaffe, te og kakao. Producenten har beskrevet noget af funktionaliteten vha. flowcharts.

- Undersøg de udleverede flowcharts, og find ud af deres funktion, inklusive deres indbyrdes interaktion.
- Skriv funktioner der håndterer valg af drik, betaling og simulering af påfyldningen. Tag udgangspunkt i den udlevere GUI.
- Udvid programmet med funktioner, som du synes er passende for en kaffeautomat.

Eksempel 8

Udleveret program hvor der er mulighed for at søge på musikkunstner og få informationer om musiknumre.

- Undersøg koden, redegør for programmets opbygning og lav et klassediagram der beskriver programmet.
- Vælg en eller flere af funktionerne i programmet og omskriv funktionen. Begrund dine valg i den nye funktion fx ud fra overvejelser om sikkerhed eller kompleksitet.
- Beskriv en eller flere nye funktioner der kunne tilføjes programmet og implementer evt. en af disse.