[Uppgift 1 - Tre “enkla” programmeringsuppgifter]

Planerad tid för alla tre uppgifter: 1 h 45 min

Strategi:
Det här är ett helt nytt upplägg för mig att arbeta med.

Min planeringsstategi är först att läsa igenom uppgifterna som ska utföras. Jag kommer sedan att kolla på kod som jag tidigare har. Därefter kommer jag att söka information på områden man inte har koll på (referenser från böcker, msdn, stackoverflow och andra sidor man stöter på, om jag hittar bra referenser noterar jag det.

När jag börjar koda så ska jag försöka notera de problem som förekommer. Jag försöker göra allt så noggrant och tydligt som möjligt.

[1a]

Planerad tid: 30 min
Det är inte så mycket planering som behövs. Kolla upp referenser på hur man kan räkna upp specifika bokstäver i en sträng, kolla lite på koder som man har skrivit i c# kurser (friska upp minnet).
Sedan skriva koden och löser eventuella problem som man stöter på. Tänka på hur man kan förbättra koden man skriver. Funderar på om man ska implementera dem eller inte.

Förbrukad tid: 25 min

Orsaker till avvikelser:
Bara 5 minuter och det var för att det var lite enklare än vad jag trodde det skulle vara samt att inga problem förekom (enkel kodning utan tillägg).

Problem:
Inga problem uppstådde, anledningen till det var att programmet var enkel/simpel/kort. Inga felhantering, loopar eller så.

[1b]

Planerad tid: 30 min
Samma struktur på planeringen som förra uppgiften.
Uppgiften verkar vara lik förra uppgiften RaknaA, liksom omfattningen är inte så stor.
Om det är så att jag kan använda samma format så kommer jag att testa det. Sedan så ska jag ta ytterligare mer tid för att söka efter information på andra sätt.

Förbrukad tid: 35 min

Orsaker till avvikelser:
Inte nöjd med första koden. Det var inga problem som förekom men jag tyckte att det blev mer kod än det vad behövdes för uppgiften.
Så jag letade efter andra sätt och hittade då några olika varianter (vissa som man inte förstår och andra som man förstår). Jag tyckte att det blev smidigare med modulus operatorn.

[1c]

Planerad tid: 45 min
Samma struktur på planeringen som förra uppgiften.
Uppgiften verkar vara svårare än tidigare uppgifter, så jag planerar att det kommer att ta ca 45-60 minuter för arbetet.

Förbrukad tid: 67 min

Orsaker till avvikelser:
Slö start (**mycket tid åt funderingar**), hantera problem och snygga till koden.

Problem:

#1

Tankar på hur jag ska sätta dessa if (? > ?), else if (? > ? && ? < ?).
Testade först

```
(input > sLargest)
largest = input
```

(vilket stämde inte när man sedan klottrade på papper för att testa tankarna)

Testade även med

```
(input < largest && input > temp) // i storleks ordning 1, 2 och 3 är temp 3
sLargest = input
```

(funkade inte heller, för input kunde vara mindre än sLargest)

#2

Körde med en for-loop men det gick inte så bra när man gjorde en felinmatning så jag gick över till en while-loop och fick et att fungerar.

Ett annat problem som förekom efter bytet var att när jag matade in 1, 2, 3, 4, 5, 6, 7, 8, 9 och sedan 10 fick jag resultatet 8 istället för 9.
För att lösa det ändrade jag while (true) med break till while (count < amountOfTimes).

#3

Det dyker upp små problem då och då när man gör testerna regelbundet men dessa löses väldigt snabbt med små ändringar. Till exempel då programmet loopar en extra gång, utmatningen (t.ex då count börjar på 0).

[Reflektioner för uppgift 1]

Jag förväntade mig att jag skulle förbruka mer tid än vad jag hade planerat, vilket var precis det som hände. Gick över med en kvart pga uppgift 1c.

Anledningen till att det tog extra tid var på grunda av att jag inte visste hur jag skulle börja. Liksom vilka och hur många variablar behöver jag för att hantera jämförelserna.
Fick inte använda arrayer. Behöver jag 10 integers för varje inmatning, hur jämför jag dem på bästa sätt och annat. Jag fick tänk på hur jag skulle göra. Googla för att få mer insikt på vad som behövs/ kan användas för att ta reda på "näst största" talet.

Jag fick insikt på hur pass dålig min planering var. Jag borde ha planerat mer steg/moment och dela upp "programmet" mer. Liksom dokumentera vad och vilka funktioner programmet ska ha. Liksom redan under planeringen ta reda på hur programmet ska uppföra. Hur den hanterar felinmatning, hur utskriften ska se ut.

Jag får tänka på hur jag lägger upp tiden för arbetet. Liksom vilka dagar och hur mycket tid ska jag lägga ner för det. Ska jag ta och köra allt på en gång? eller ska jag fördela arbetet över veckan. Det jag har gjort är i princip arbeta när jag känner för det, vilket blev mycket fritid. Det blev lite tajt de dagarna före redovisningen. Jag kände och visste att det var inte själva kodningen som skulle ta tid utan skriva dessa reflektioner och planeringar.

Tycker att planeringen av tiden var inte illa, gick över med en kvart (anlednin till avvikelse har jag nämnt).

Bristen på information och vagt ställda krav kring uppgifterna skapade en viss osäkerhet tyckte jag . Om uppgifterna hade mer specifika krav och lite mer information skulle det bli bättre. Liksom nämna med eller utan felhantering, loopar eller så. Så behöver man inte ställa den frågan för sig själv (om man ska implementera dem eller inte, mer övning men "behöver" jag verkligen lägga ner tid för det?).

Förutom tid åt planering och kodning så har mycket tid spenderat för dokumenteringen av arbetet. Ska kanske dokumentera tiden för detta också.

Jag är inte så bra på att planera mitt arbete men jag ska "försöka" göra mitt bästa att bli bättre på det, saknar lite vägledning förtillfälle. Liksom tips och råd är skönt att ha med sig. Sen så förstår jag att på ett professionellt sätt måste man planera, vara strukturerad och ha en överblick över sitt arbete. Jag känner att den här kursen är en bra möjlighet att lära mig en massa nyttig information som jag kommer ha nytta av i yrkeslivet.

[Uppgift 2 - Förändring och förbättring]

[a]

En strategi som jag kommer att implementera för mitt arbete är att under planering redan då göra en liten skiss för koden. Först verkligen se till så att man förstår vad det är man ska göra sedan visualisera

[b]

Det verkar vara svårt att ta hänsyn till okända variabler. Det man kan göra är nog att gå igenom de uppgifterna man ska göra "noggrannt" och så att man "förstår" det man ska göra. Då har man en bättre koll på vad man ska och inte göra, vilket underlättar arbetet. Istället för att lösa uppgifterna steg för steg tycker jag att det är bättre med att först gå igenom hela och sedan såklart dela upp momenten i flera steg.

Genom att ha en bra förståelse för det man ska arbeta med, får man en bättre koll med tidsåtgången för eventuella fel och alla andra saker som inträffar.
Då kan man räkna in den tiden för den planerade tidsåtgången.

Det är nog bättre att sätta den planerade tidsåtgången för "mer" istället för "mindre".

[c]

1. Under planeringen ska jag tänka igenom de olika steg som ska utföras för att lösa uppgiften och sedan **skissa ut en grov algoritm för koden.** Med skissen får jag en bättre insikt för **tidsåtgången** och det är pågrund av mer information. Här planerar jag in eventuella problem/fel små eller stora.

2. **Bättre och mer noggrann dokumentering av fel och avvikelser.** Jag kommer att växla mellan kodningen och dokumenteringen istället för att dokumentera "efter" man är klar med kodningen. Genom att växla mellan kodning och dokumentering så får jag en mer detaljerad och noggrann "rapport".

Dock så kan det finnas nackdelar med detta eftersom man kan glömma tankar man hade för kodningen då man byter växel när man växlar mellan den ena till den andra. För det ska jag försöka med att **kommentera** tankarna ifall man då ska "växla".

[Uppgift 3 - Förbättrad planering av programmering]

Planerad tid för båda uppgifter: 2 h
Jag kände att mina erfarenheter räckte inte för att lösa dessa uppgifter smidigt, för att utföra dem bättre behöver jag friska upp minnet och läsa mera.

Strategi:
Min planeringsstategi utöver det tidigare är då först att läsa igenom uppgifterna som ska utföras och förstår vad det är som ska utföras. Jag kommer dessutom att göra en liten skiss av koden innan jag börjar med kodningen (så att jag har en smidigare "mall" att gå efter). I samband med skissen kommer jag att söka information på områden man inte är så säker på.

Jag kommer att notera de problem som förekommer och de som "kan" förekomma.

Jag kommer testa en annorlundare format.

[3a - Planering]

Tid: 1 h

Språk: C#

Hjälpmedel: Visual studio

Hur programmet ska fungera:

- Be användaren mata in textrad
    - Om det finns stora bokstäver
        - Felmeddelande
            - Ny chans (ny inmatning)
    - Inga stora bokstäver
        - Kontrollera om ordet är Palindrom
            - Är palindrom
                - Meddela svaret
                    - Slut
            - Är inte palindrom
                - Meddela svaret
                    - Ny chans (ny inmatning)

[3b - Planering]

Planerad tid: 2 h

Språk: C#

Hjälpmedel: Visual studio

Notiser:
- En konstruktor som skapar och initialiserar ett nytt bråktal.
- isEqualTo som jämför två Fraction-instanser och ser om de representerar samma bråktal.
- toString som returnerar en strängrepresentation av bråktalet på form T/N.

Algoritm:
- Skapa klass "Fraction"
- Skapa privata fält för täljare och nämnare (int)
- Skapa konstruktor för ett nytt bråktal
- Skapa metoden getNumerator
    - Returnerar täljaren
- Skapa metoden getDenominator
    - Om nämnaren är noll
        - Felmeddelande
    - Returnera nämnaren
- Skapa metoden isNegative
    - Returnerar "true" om det är ett negativt bråktal
- Skapa metoden add
    - ?
- Skapa metoden multiply
    - ?
- Skapa metoden isEqualTo()
    - Jämför två Fraction-instanser och ser om de representerar samma bråktal.
        - return true ?
- Skapa override toString()
    - Returnerar "T/N"

* ? = osäker område

[Uppgift 3 - Reflektioner]

[a] Planerad tid: 1 h Verklig tid: 41 min Skillnad: 19 min

Eftersom jag ärligt talat inte visste vad en palindrom var så tog jag tid för att läsa om det och hur man kan kontrollera om en sträng är en palindrom eller inte. Det jag fick reda på att man kan konvertera den inmatade strängen "bakvänd" och sedan jämföra om den inmatade strängen och den bakvända srängen är lika.

Orsaker till avvikelser:

Det gick smidigare än vad jag hade planerat eftersom det var enklare än vad jag trodde det skulle vara. Jag behövde bara läsa om vad och hur en palindrom fungerar, hur man kontrollerar om en sträng är en palindrom eller inte. Läsa mera om IsUpper, läsa om hur man vänder en sträng. Sedan var det ganska enkelt med att skriva koden.

Problem:

Inga större problem, bara små fel på hur man satte if och else satserna. Färghantering och radbrytningar. Fixa loop så det beter sig på rätt sätt.

[b] Planerad tid: 2 h Verklig tid: 3 h Skillnad: 1 h

Denna uppgift såg ut att vara mer omfattande än tidigare uppgifter, därför bestämde jag en längre tid. Det tog mycket mer tid än vad jag hade planerat.

Orsaker till avvikelser:

Det tog en hel del extra tid eftersom man helt enkelt inte var säker på programmets funktion. Det nämndes inte om vad programmet ska göra utan bara att man ska skapa klassen "Fraction" och implementera de olika medlemmarna. Jag visste inte hur add, multiply och isEqualTo skulle fungera, främst isEqualTo().

Det som tog mest tid var då hur inmatningen och utskriften skulle se ut, jag testade ett par olika sätt och det tog en bra tid tills jag blev nöjd med den.

Jag hade inte en så bra koll på hur jag skulle testa metoderna add och multiply men det klurade jag ut. Jag testade dem på ett simpelt sätt, jag försökte tidigare se till så att man kunde mata in egna bråktal men jag fick inte till loopen och felhantering för detta. Försökte med en loop för två bråktal, där om man skriver in t.ex fel vid "mata in täljare" ska man få en ny chans för "mata in täljare". Istället gick den vidare till "nämnaren" och sedan till bråktal två. Ville inte ha för mycket kod här, annars så skulle det förmodligen gå, men det blev då "DRY".

Problem:

Problem var då kring hur loopen/looparna skulle bete sig vid felinmatning. Sen så fanns det slarvfel men de var inte så omfattande. Andra problem var då att jag vid början inte visste hur vissa metoder skulle fungera. Så det blev "mycket" tankar och funderingar.

[Uppgift 4 - Planering]

CloudPortfolio

Notiser:

- Lagra och dela dokument på molnet

Funktionalitet:

- Loggar in - Visar filer och mappar.
- Filer - Färgkodning beroende på vem som har redigerat den:
    - En färg för sig själv.
    - Varje person har olika färger.
    - Användaren som utfört ändringen visas.
    - Vilken typ av modifiering visas.
- Användaren har en egen *portfolio*.
    - Kan lägga till dokument:
        - Kan ladda upp befintliga OOXML formaterade dokument.
        - Kan skapa dokument via inbyggd textredigerare.
    - Kan flytta, byta namn och ta bort dokument och mappar.
    - Dokument har en åtkomstkontrollista kopplad till sig.
        - Kontrollerar ägandet och nyttjanderätten.
        - Endast ägare och administratörer kan ändra åtkomsten.
        - Kan välja att dela ett dokument eller en mapp med andra användare.
            - Isåfall kommer ägaren att tilldela rättigheter.
                - En inbjudan skickas till användaren.
- Användaren kan ta bort sitt konto.
    - Alla filer och mappar som är kopplat till kontot raderas.
    - Har användaren delat administrativa rättigheter med andra så kan de tas bort endast om alla är överens.
        - Skicka ett meddelande de omfattade användare.
            - Om alla är överens så raderas filerna.
            - Om någon inte samtycker kommer filerna inte att tas bort.
        - All användarinformation raderas.

Planering:

- 4 veckors iterationer.
- v 1:
    - Möte - Diskuterar och målar upp en sketch för projektet.
    - Kontrollera resurser.
    - Alla tar del av sketcherna.
    - Gruppfördelning (planering inom grupperna utförs).
        - Grupp A - Fokus på filhanteringen och databasen.
        - Grubb B - Fokus på webbsidan (html, css), inloggning och textredigerare
    - Börja koda och testa.
- v 2:
    - Fortsätt med samma arbete.
- v 3:
    - Fortsätt med samma arbete.
    - Grupperna får ha möten för att förklara vad som har uppnåtts.
    - Beroende på hur färdig eller inte färdig applikationen är:
        - Tung testning.
        - Börja de-scopa iterationen om man inte kan nå målet.
- v 4:
    - Incheckning och kod frysning för iterationen.
    - Möte för planering av nästa iteration.

- Andra månaden:
    - Om applikationen är färdig:
        - Tung testning.
    - Om applikationen är inte färdig:
        - Fortsätt med samma arbete.

- Tredje månaden:
    - Fortsätt som tidigare.