**Oppgåve 1B**

Enhetstesting er ein metode for å sjekke om dine funksjonar faktisk returnerer det du forventar. Dette blir gjort ved at ein køyrer funksjonane med forskjellige input, og kontroller om den kvar gong gjev det svaret som ein forventar. Fordelen med enhetstesting er at du i tur og orden får sjekka kvar funksjon som du skal bruke på ei nettside. På den måten er det enklare å finne feilen dersom sida di ikkje fungerer slik som den skal. Det kan vere vanskeleg å nøste opp i feilen som skjer dersom du berre prøver å køyre ei nettside, og resultatet ikkje blir som du hadde trudd. Ein liten ulempe med enhetstesting er at det tar litt tid å setje opp, men etter at det er sett opp tek sjølve testinga liten tid. Og ein liten fare med enhetstesting er at du slurver når du skriv sjølve testen. Dette kan føre til at testen feilar sjølv om funksjonen din faktisk fungerer som den skal, og på den måten blir du kanskje sitjande for å prøve å rette ein feil som ikkje eksisterer.

Eg har så vidt prøvd enhetstesting tidlegare, men har så å seie ingen erfaring med det frå før. Eg føler at å programmere med enhetstester fekk meg i enda større grad til å tenke over korleis eg kunne skrive funksjonen ekstra sikker. Skal innrømme at eg for eksempel sikkert kunne gløymt at ein bruker av sudoku-spelet kunne finne på å skrive inn ein bokstav eller eit anna teikn som ikkje høyrer heime der.

**Oppgåve 2B**

Model view controller er eit designmønster der koda er delt i tre delar som har kvar sin “oppgåve”. I *model* ligg variablane som inneheld informasjonen om kva som skal bli vist på nettsida. Frå koda i innleveringsoppgåva så er eit eksempel på dette variabelen som inneheld sudokubrettet – kva tal som faktisk kjem opp brettet når sida blir lasta inn og etter kvart når sida blir oppdatert ved at vi tastar inn nye tal. Den andre delen, *view,* styrer grensesnittet. Denne delen består altså av funksjonar som styrer korleis informasjonen i modellen faktisk blir vist på nettsida. Funksjonane som styrer grensesnittet tek informasjonen sin frå modellen, og presenterer den på den måten som vi ynskjer. Det at tala i sudokubrettet faktisk blir vist i eit sudokubrett, og ikkje for eksempel berre som ei liste med tal, er det desse funksjonane som styrer. Det er også desse funksjonane som gjer at bakgrunnen på brettet blir raud dersom vi skulle taste inn eit feil tal i sudokubrettet. *Controller* kan vi si er bindeleddet mellom *model* og *view*. Dette er funksjonar som tek inn informasjon frå brukaren av nettsida (inntastingsfelt, radioknappar eller liknande), og som på grunnlag av dette oppdaterer variablane i *model*. Etter at funksjonane som skal oppdatere éin eller fleire av variablane har gjort sin jobb, blir funksjonane frå *view*-delen køyrt på nytt. På denne måten blir heile nettsida oppdatert, og brukaren av nettsida har ofte moglegheit til å sende inn ny informasjon som gjer at heile prosessen skjer igjen. Eit eksempel frå sudoku-spelet er funksjonen som tar inn tala som spelaren tastar inn. Denne funksjonen har da som oppgåve å oppdatere variabelen som inneheld tala på brettet. Denne funksjonen vil også oppdatere variablane som inneheld informasjon om spelaren har gjort ein feil, eller om spelaren er ferdig. Deretter blir funksjonane frå *view* køyrt og vi får opp sudokubrettet på ny med den nye informasjonen (det nye talet).