Eksamen

Dette dokumentet beskriver oppgaveteksten for eksamen i Innføring i Programmering under høstsemesteret 2023 ved Høgskolen i Østfold.

Eksamen er en mappeoppgave bestående av tre mappebidrag, kalt oppgave 1, 2 og 3, i tillegg til et mappedokument. Alle oppgavene har minstekrav som må bestås for å få godkjent eksamensmappen. Hvert mappebidrag er beskrevet videre i dette dokumentet.

Tips til oppgavene

Les først dette dokumentet nøye. En god oversikt over forventede løsninger vil gjøre arbeidet betydelig lettere.

Sett deg godt inn i HTML-en som gis i oppgavene. Her vil det være nyttige selectorer og strukturer som kan gi en pekepinn på hvordan noen av oppgavene kan løses. All HTML-en fungerer til og med A-kravene i oppgavene. Det er altså ikke forventet at dere skal endre/skrive HTML og CSS i eksamensoppgavene.

Løs en liten bit av gangen. Ta et og et krav, og gjerne bit for bit av kravet. Algoritmisk tankegang, tegning og planlegging vil være den største hjelpen på vei mot løsningene.

OBS: Du kan lage egen HTML, men da stilles det krav til at denne kommenteres og at struktur og planlegging dokumenteres i mappedokumentet!

Mappedokument

Mappedokumentet **skal** inneholde

- Hvilket karakterkrav du har tatt sikte på å nå for hvert av mappebidragene
- Informasjon/grundig beskrivelse av hjelp fra medstudenter eller KI-tjenester som ChatGPT
- Referanser/kilder du har benyttet under arbeidet. Referanser og kilder til spesifikke kodebiter (eksempel: en funksjon du har funnet på StackOverflow og brukt i koden din med mindre tilpasninger) kan med fordel også refereres direkte i koden som en kommentar.

Mappedokumentet **kan** også inneholde følgende for å telle positivt mot evaluering

- Planleggingsskisser/notater og dokumenter som viser algoritmisk tankegang rundt problemløsning av mappebidragene.
- Redegjørelse for endringer/egen utvikling av HTML-kode utover HTML-kode som er gitt med oppgavene (se avsnitt «Tips til oppgavene» over)
- Eventuell informasjon nødvendig for sensor (eksempelvis dersom du nesten har løst et krav i et av mappebidragene og har en idè om hvordan det løses, men ikke har fått tid til å feilsøke/utforske det nok til levering)

Oppgave 1: Kalkulator (vekt: 20%)

Denne oppgaven tester feilsøking, forståelse av funksjoner og utskrift til grensesnittet, samt bruken av arrayer og interaksjon med brukerne i de høyere karakter-oppgavene.

I mappen oppgave1 ligger en fil kalt eksamen_oppgave1.html. Åpne denne i en editor og i nettleseren.

Mål med oppgaven

- Fikse feil slik at kalkulatoren fungerer
- Vise forståelse for kodesammenhenger
- Lage en memory-funksjon hvor brukeren kan lagre mattestykker på skjermen, og hente disse fram igjen ved å bla frem og tilbake i memory-banken. OBS: I memorybanken kreves det kun at mattestykket som er aktivt i skjermen når brukeren klikker add-knappen lagres. Det er ingen krav til at disse skal kombineres med andre utregninger, eller huske andre steg i utregningen enn det som synes i skjermen når brukeren klikker add
- Memory-funksjonalitet:
    - Når brukeren klikker knappen Add, lagres det mattestykket som vises i skjermen i kalkulatorkoden.
    - Når brukeren klikker Previous-knappen, henter kalkulatoren fram det siste (nyeste lagrede) mattestykket fra memory. Dersom det er flere lagrede mattestykker, skal ytterligere klikk på Previous-knappen hente fram et og et eldre mattestykke fra memory.
    - Når brukeren klikker Next-knappen, henter kalkulatoren fram det neste lagrede mattestykket i memorybanken i forhold til det aktive mattestykket.

Next-knappen kan med andre ord ikke brukes før Previous-knappen er brukt, og det finnes mer enn et mattestykke i memory.

- Previous- og Next-knappene er ikke tilgjengelige (disabled) ved oppstart. Disse må gjøres tilgengelig for brukeren når de skal brukes.
- Previous og Next-knappene bør også gjøres utilgjengelige når de ikke gir noe memory, altså dersom vi viser det første lagrede mattestykket i memory vil det ikke finnes noe previous, og hvis vi viser det siste mattestykket i memory vil det ikke finnes noe next. Husk at et mattestykke i memory kan være både det første og det siste.

Se også video av hvordan resultatet av oppgaven skal se ut (inkluderer til og med A-krav): https://hiof.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=b6119a93-8d7b-4681-80ec-b0ab00d0e911

Minstekrav (E-krav)

1. Bruk konsollen og løs de fem feilene som sørger for at kalkulatoren fungerer.
2. Skriv en kommentar i koden under solve()-funksjonen som grundig forklarer hvordan solve()-funksjonen fungerer.

D-krav

1. Lag funksjonalitet som lagrer mattestykket som står i skjermen når du klikker Add-knappen i Memory-funksjonen.
2. Sørg for at tallet i parentes bak overskriften memory oppdateres i henhold til hvor mange mattestykker som er lagret i memory.
3. Sørg for at det som ligger lagret i memory skrives ut til konsollen for kontroll.

C-krav

1. Lag funksjonalitet som henter ut det siste lagrede mattestykket i memory ved å klikke Previous-knappen, og skriv dette mattestykket til skjermen.
   OBS: Hvis du ikke ønsker å gjøre B- og A-kravene i denne oppgaven, endre knappeteksten på Previous-knappen til «Fetch from memory», og fjern Next-knappen fra HTML-en.

Tips/forslag til A og B-krav

- Ved klikk på Add-knappen, når bruker legger til et nytt mattestykke i memory, kan det forutsettes at man alltid henter det nyeste (sist lagrede) mattestykket fra memory når man trykker previous

B-krav

1. Lag funksjonalitet som gjør det mulig å hente eldre mattestykker fra memory ved å trykke flere ganger på previous-knappen.
2. Sørg for at previous-knappen blir utilgjengelig når det ikke er flere mattestykker og bla tilbake i.

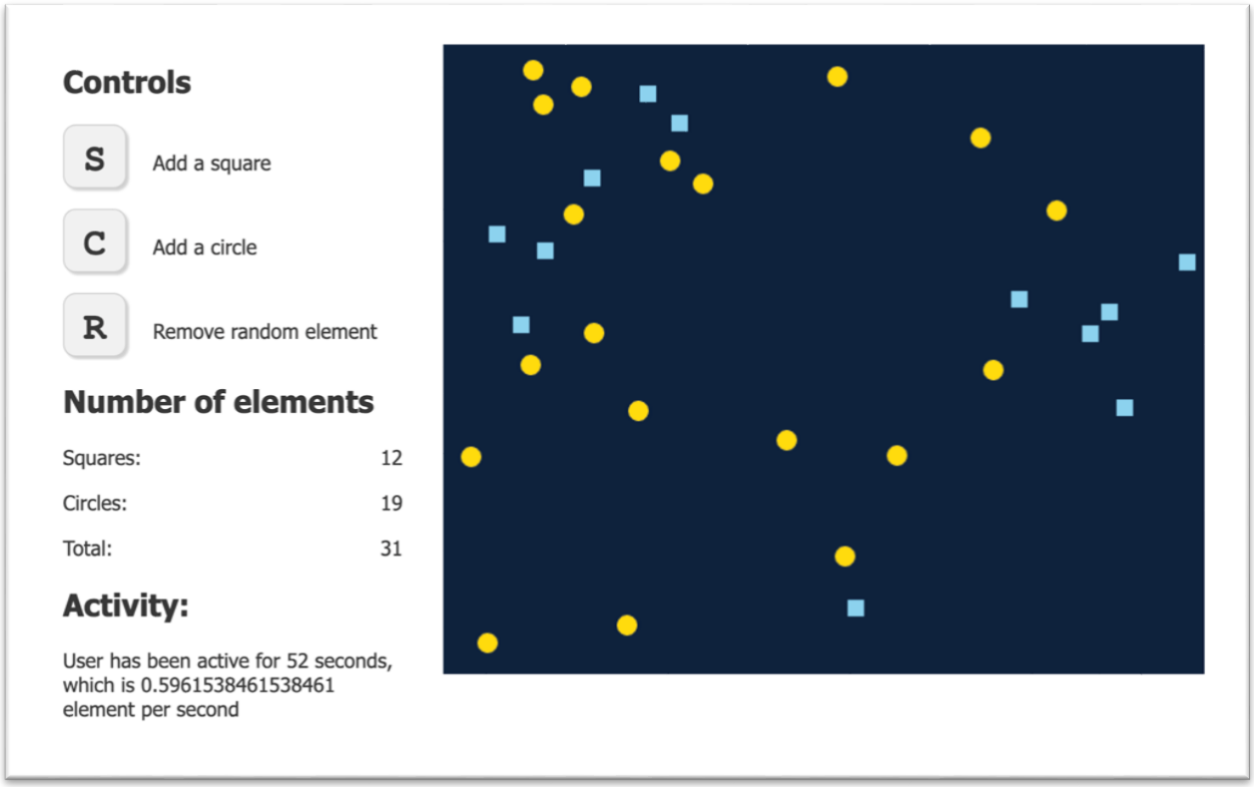
A-krav

1. Lag funksjonalitet som gjør det mulig å bla framover i mattestykkene i memory med Next-knappen etter det er bladd tilbake med Previous-knappen.
2. Sørg for at previous-knappen blir utilgjengelig når det ikke er flere mattestykker å bla fremover i.

Oppgave 2: Randomisert tegning på canvas (Vekt: 30%)

Denne oppgaven tester konseptene tegning i canvas, funksjoner, randomisering og arrayer.

I mappen oppgave2 ligger en fil kalt eksamen_oppgave2.html. Åpne denne i en nettleser og en editor.



Mål med oppgaven

- Når man klikker på S-knappen i grensesnittet eller S-tasten på tastaturet, skal det tegnes en blåfarget firkant (square), 8 x 8 pixler stor, på en tilfeldig plass innenfor canvaset.
- Når man klikker på C-knappen i grensesnittet eller C-tasten på tastaturet, skal det tegnes en gullfarget sirkel (circle) med en radius på 13 pixler på en tilfeldig plass innenfor canvaset.
- Når man klikker på R-knappen i grensesnittet eller R-tasten på tastaturet, skal en tilfeldig valgt firkant eller sirkel fjernes fra canvaset.
- Grensesnittet skal oppdateres med en opptelling av antall firkanter, antall sirkler og antall elementer totalt ettersom flere elementer legges til eller fjernes fra grensesnittet.

Se også video av hvordan resultatet av oppgaven skal se ut (inkluderer A-krav):
https://hiof.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=3f8590e0-f5b7-4f15-9377-b0ab00ca975c

Minstekrav (E-krav)

Minstekravet må være bestått for at oppgave 2 skal telle som bestått.

1. Museklikk på S-ikonet i grensesnittet tegner en firkant som beskrevet i mål med oppgave over i canvaset ved hjelp av JavaScript
2. Museklikk på C-ikonet i grensesnittet tegner en sirkel som beskrevet i mål med oppgave over i canvaset ved hjelp av JavaScript
3. Tellerne oppdateres når det tegnes et element i canvaset
4. Kode du produserer kommenteres

D-krav

1. I tillegg til museklikk på S-ikonet i grensesnittet skal også trykk på S-tasten på tastaturet tegne en firkant som beskrevet i mål med oppgave over i canvaset ved hjelp av JavaScript
2. I tillegg til museklikk på C-ikonet i grensesnittet skal også trykk på C-tasten tegne en sirkel som beskrevet i mål med oppgave over i canvaset ved hjelp av JavaScript

C-krav

1. Ytterligere klikk på S- eller C-alternativene fra de tidligere kravene legger til nye elementer, henholdsvis firkanter og sirkler, på tilfeldige plasseringer innenfor canvaset.
2. Tellerne som viser antall elementer oppdateres ettersom nye elementer legges til.

B-krav

1. Museklikk på R-ikonet i grensesnittet eller R-tasten på tastaturet fjerner et tilfeldig valgt element fra canvaset.
2. Tellerne som viser antall elementer oppdateres ettersom elementer fjernes.

A-krav

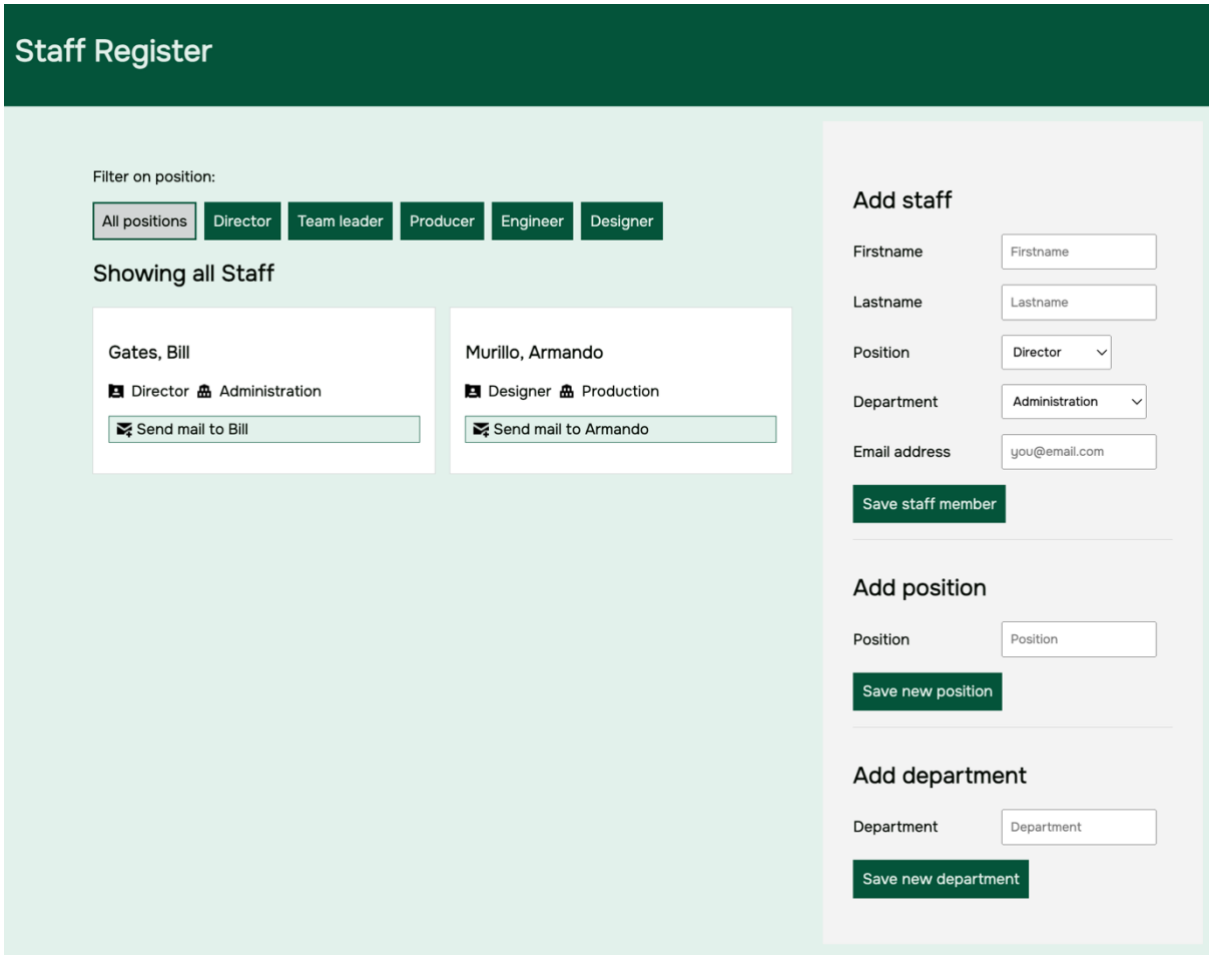
1. En teller oppdateres og viser hvor lenge spilleren har vært aktiv på nettsiden hvert sekund.
2. En gjennomsnittsutregning av hvor mange elementer som finnes per sekund brukeren har vært aktiv oppdateres hvert sekund.

Oppgave 3: Ansattregister (Vekt: 50%)

Denne oppgaven fokuserer på planlegging, sammenheng mellom funksjoner, interaksjon med grensesnittet og arrayer.

I mappen oppgave3 ligger det to filer:

- register.html: dette er filen du skal skrive kode i
- register_HTML.html: I denne filen ligger det fullstendig HTML-kode som eksempler for hvordan filter-navigasjon, visning av ansatte og select-bokser (nedtrekksmenyer) i skjemaet skal se ut etter JavaScript har generert det. Merk: det er ikke noe JavaScript-funksjonalitet i denne filen, så grensesnittet gjør ingenting, dette er kun så dere har HTML-strukturen tilgjengelig for videre utvikling av register.html.



Mål med oppgaven

- Du kan registrere nye ansatte i ansattregisteret med fornavn, etternavn, stilling, avdeling og epostadresse. **MERK**: De ansatte skal ikke lagres i noen database, kun i koden du skriver i denne filen. Det vil si at endringer du registrerer vil forsvinne når siden lastes inn på nytt.
- Du skal ikke kunne registrere nye ansatte dersom noen av feltene fornavn, etternavn eller epostadresse er tomme.
- Du skal kunne legge inn nye stillinger og avdelinger, som etter de er opprettet skal kunne velges når du oppretter en ny ansatt.

- Ansatte skal kunne filtreres på stilling.
- Grensesnittet markerer hvilken stilling det er filtrert på gjennom markering i filternavigasjon og ved endring av tekst i heading for ansattliste.

Se også video av hvordan resultatet av oppgaven skal se ut (inkluderer A-krav: https://hiof.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=b6119a93-8d7b-4681-80ec-b0ab00d0e911 .

Minstekrav (E-krav)

Minstekravene må være møtt for at oppgave 3 skal telle som bestått

1. Ansatte skal listes opp i ansattlisten
2. Du skal kunne registrere en ny ansatt gjennom HTML-skjemaet. Nye ansatte lagres i koden.
3. Den nye ansatte skal vises i ansattopplistingen så fort denne er registrert
4. Kode du produserer er kommentert

D-krav

1. Stillinger og avdelinger i nedtrekksmenyene i HTML-skjemaet skal sorteres alfabetisk fra A-Å
2. Nye avdelinger kan opprettes via HTML-skjemaet
3. Avdelingen blir tilgjengelig i nedtrekksmenyen for valg av avdeling i HTML-skjemaet for registrering av ny ansatt så fort den er opprettet

C-krav

1. Ansattvisningen skal sorteres på etternavn
2. JavaScript genererer filtermenyen over ansattlisten basert på stillinger
3. Nye stillinger kan opprettes via HTML-skjemaet
4. Stillingen blir tilgjengelig i nedtrekksmenyen for valg av stilling i HTML-skjemaet for registrering av ny ansatt så fort den er opprettet
5. Stillingen blir tilgjengelig i filtermenyen over ansattlisten som et filtervalg så fort den er opprettet.
6. Valget «alle» er fortsatt tilgjengelig i filtermenyen etter den er oppdatert med ny stilling.

B-Krav

1. Klikk på et menypunkt i filtermenyen filtrerer ansattlisten korrekt for alle stillinger.
2. Ansattlisten viser kun ansatte som matcher valgt filter, og oppdateres straks menypunktet i filtermenyen klikkes.
3. Overskrift over ansattlisten oppdateres med antall ansatte som matcher filtreringen, og hvilken stilling det er filtrert på, ala «Showing 1 employee with position Designer»
4. Valget «all» i filtermenyen tilbakestiller ansattlisten og viser alle ansatte alfabetisk sortert på etternavn

A-krav

1. Filtermenyen markeres basert på hvilket filter som er valgt.

Leveranse

Du skal levere en komprimert (zippet) mappe som består av tre mapper, oppgave1, oppgave2 og oppgave3. Disse tre mappene skal inneholde dine løsninger på eksamensoppgavene. I tillegg skal den komprimerte mappen inneholde et mappedokument, som beskrevet i kapitlet om mappedokument.

Leveransen er individuell, og ethvert arbeid som ikke er løst individuelt skal beskrives i mappedokumentet.

Mappen skal leveres i eksamensoppgave for Innføring i Programmering i Inspera innen fredag 8. desember 2023 klokken 14:00.