Tentamen för kursen TME136 Programmering och algoritmiskt tänkande

Tid: 29 oktober 2020 kl 14:00-18:00

Lärare: Henrik Ström

Lärarkontakt: Om du har en fråga till examinatorn, skriv i Zoom-chatten till "Everyone": *Fråga till examinatorn*. Tentavakten kommer att ändra ditt namn och lägga till ##EXAMINATOR## framför. Du kommer att bli flyttad till examinatorns rum i Zoom för att ställa din fråga. Det kan dröja en stund om examinatorn är upptagen eller tillfälligt otillgänglig. Notera att detta är det sätt på vilket kontakter med examinator normalt ska ske. Om detta av någon anledning inte fungerar, eller i händelse av extraordinär situation, går examinatorn också att nå på mail (henrik.strom@chalmers.se) eller telefon (031-7721360). Kontakt på detta sätt måste först stämmas av med tentavakt för att kunna avfärda olovlig kontakt med annan utomstående!

Tillåtna hjälpmedel: Alla. Samarbete är ej dock ej tillåtet, vare sig med annan tentand eller utomstående.

Formalia: I programhuvudet på respektive m-fil ska du skriva in ditt namn.

Programmen du skriver ska vara tydliga och enkla att följa med förklarande kommentarsatser, och det ska vara enkelt att göra rimliga mindre ändringar. Detta innebär att hårdkodade lösningar i normalfallet inte kan förväntas ge full poäng.

Det är tillåtet att skapa nya filer utöver de som redan finns, de bör i så fall förses med lämpliga programhuvuden. Programfilerna till respektive uppgift finns och ska efter bearbetning sparas i tillhörande mapp Uppgift1-Uppgift5. Du får bara lämna en lösning till varje uppgift. En delvis löst uppgift kan ge poäng.

När du är klar med tentamen, packa ner alla filer i ett ZIP-bibliotek och ladda upp i Canvas.

Rättning: Resultatet anslås senast torsdagen den 19:e november 2020 i Canvas. Det kan också ses i Studentportalen och Ladok (eventuellt med viss fördröjning). Granskning sker onsdagen den 25:e november kl 12.30-13.15 via Zoom (länk postas i tentans Canvas-rum).

Betygsgränser: Poängantalet för korrekt besvarad/löst uppgift anges inom parentes (p). Betygsgränser för tentamen är:
Betyg U $<$ 16p ; 16p $\leq$ Betyg 3 $<$ 24p ; 24p $\leq$ Betyg 4 $<$ 32p ; Betyg 5 $\geq$ 32p.

LYCKA TILL!

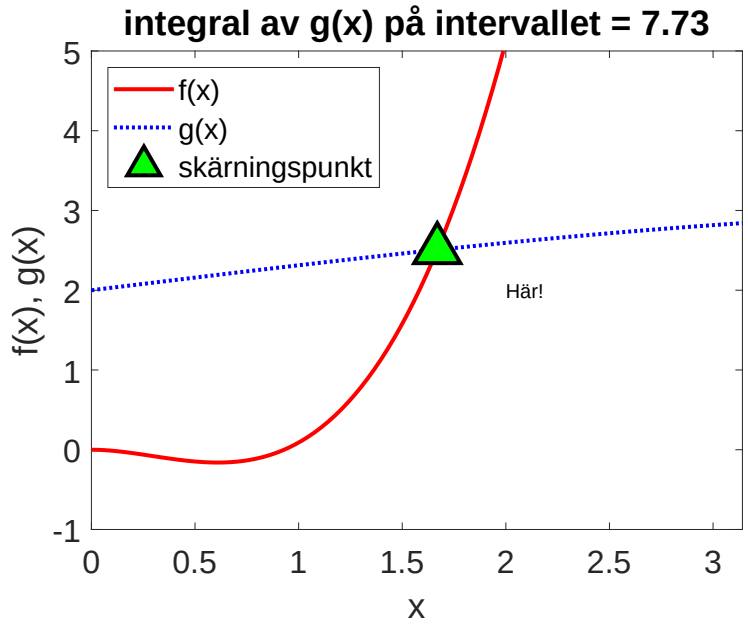
1. Småuppgifter (8p)

Uppgiften fokuserar på följande två funktioner:

$$f(x) = x^3 - \frac{x^2}{\ln(2+x)}$$

$$g(x) = 2 + sin(x/\pi)$$

Spara lösningen av följande deluppgifter i filen `Uppgift1.m` i mappen `Uppgift1`. När hela uppgiften är klar bör figurfönstret se ut ungefär som i exemplet nedan.



Figur 1: Exempelfigur för denna uppgift.

(a) Plotta $f(x)$ och $g(x)$ på intervallet $[0, \pi]$. y-axeln i figuren ska gå från -1 till 5. Linjen ska vara heldragen röd för f och prickad blå för g, och linjetjockleken ska för båda vara 2. Figurens axlar ska namnges på lämpligt sätt. Fontstorleken i figuren ska vara 16. (2.5p)

(b) Ta fram koordinaterna för skärningspunkten där f och g skär varandra. (1p)

(c) Markera skärningspunkten med en grön triangel av storlek 20 med svart kant av linjetjocklek 2. (1p)

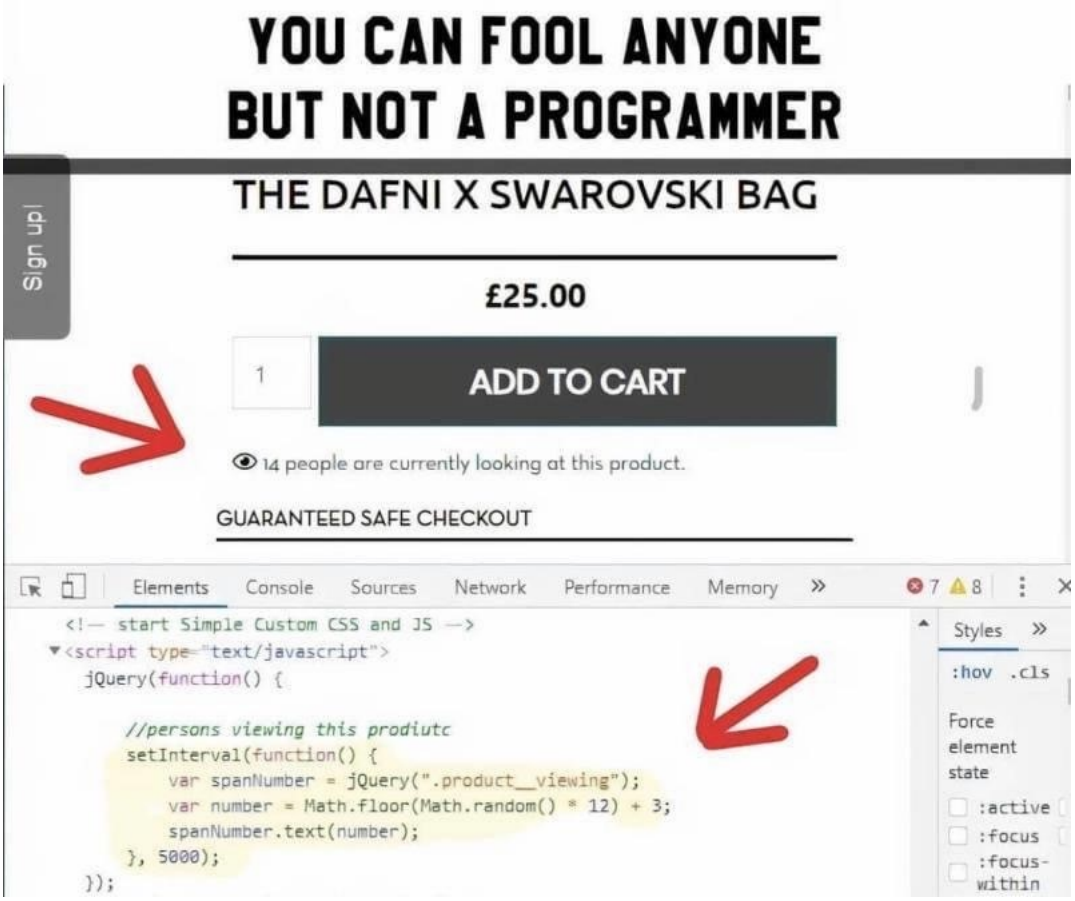
(d) Lägg till ett förklarande fönster (så kallad *legend*) i övre vänstra hörnet. (1p)

(e) Beräkna integralen $\int_0^\pi g(x)dx$ och skriv ut en rubrik i figurfönstret där det står vad integralen blev. (1.5p)

(f) Lägg till ordet *Här!* någonstans i närheten av skärningspunkten (se figur). (1p)

2. Populära väskor (8p)

Alla som handlat online vet att man ibland kan se uppgifter om hur många andra besökare som för tillfället är intresserade av att köpa samma sak, allt i syfte att stressa till impulsköp för att inte missa ett bra tillfälle. Bilden nedan visar ett exempel där man även illustrerat att den kod som tar fram antalet andra intressenter på just denna sida egentligen bara presenterar ett slumptal mellan 3 och 14.



Figur 2: Exempel på kod som genererar ett påhittat antal intresserade köpare av en fräsig väska.

Spara lösningen av denna uppgift i filen `Uppgift2.m` i mappen `Uppgift2`.

(a) I filen `bagdata.txt` finns information om namn och pris på ett antal olika väskor. Läs in data från filen på ett för uppgiften lämpligt sätt. (2p)

(b) Komplettera prisuppgifterna med en uppgift om hur många andra som just nu tittar på produkten, vilket ska ges av ett slumptal mellan 3 och 14. (1p)

(c) Skriv ut en formatterad tabell till Command Window som listar väska (med namn), pris och antal besökare, sorterad i prisordning (från billigast till dyrast). (3p)

(d) Under tabellen ska du också skriva ut det totala antalet besökare och den totala möjliga inkomsten (om varje besökare köper väskan de tittar på). (2p)

Om någonting händer med `bagdata.txt` kan du kopiera orginalet från underkatalogen `backup_kopia`.

3. Vandring i Hyrule (9p)

The Legend of Zelda: Breath of the Wild är ett *open-ended*-spel av Nintendo från 2017 som tog fem år att utveckla och har kommit att bli ett av världens mest sålda någonsin. I detta spel styr man hjälten Link på sina äventyr i kungariket Hyrule. Ett av spelets stora tjusningar är att man kan göra saker i vilken ordning som helst och att det finns så otroligt mycket att utforska i spelvärlden även utanför huvudstoryn. I denna uppgift ska vi återskapa spelgläden från *Breath of the Wild* i lite mindre skala, genom att skapa ett ”spel” där man kan flytta runt en markör på en karta.

Figur 3: Exempel på hur en komplett lösning till uppgiften kan se ut.

Spara lösningen av denna uppgift i filen `Uppgift3.m` i mappen `Uppgift3`.

Ditt eget spel ska ha samtliga nedan listade features för full poäng (delpoäng för delvis löst uppgift anges inom parentes):

- Spelpjäsen är en rund röd ifylld cirkel med svart kant av storlek 18. (1p)
- Figurfönstret har bilden `HonLJ.jpg` som bakgrund (se figur ovan). (1p)
- Figurfönstret öppnas i helskärmsläge (*fullscreen*) när spelet börjar. (1p)
- En dialog frågar spelaren efter en bokstav som indikerar vad Link ska göra. Om spelaren svarar ’N’ (eller ’n’) ska spelpjäsen flyttas ett steg nedåt på kartan. För ’U’/’u’ sker förflyttningen istället uppåt, för ’H’/’h’ åt höger och ’V’/’v’ åt vänster. Om spelaren svarar ’Q’ eller ’q’ avslutas spelet. Steglängden ska väljas så att det är ungefär 10 steg mellan kartans kanter i varje riktning. Man kan inte flytta spelpjäsen utanför kartan (sådana svar från spelaren ska ignoreras). (4p)
- Dialogen utförs med hjälp av en dialogruta (se kommandot `inputdlg`). (1p)
- När spelaren avslutar spelet stängs figurfönstret. (1p)

Om någonting händer med `HonLJ.jpg` kan du kopiera orginalet från underkatalogen `backup_kopia`.

4. Matematik (6p)

Betrakta följande ekvationssystem:

$$2x = z - y \tag{1}$$

$$2y - z = 18 \tag{2}$$

$$x + y = 27 \tag{3}$$

(a) Lös ekvationssystemet ovan med Matlab. Skriv ut lösningen (värdena på x, y och z) till Command Window så att det framgår vilket värde som hör till vilken variabel. Spara lösningen av denna uppgift i filen `Uppgift4.m` i mappen `Uppgift4`. (3p)

(b) Lös ekvationen $e^{-w} = 5w$ med Matlab. Koden som gör detta ska också skrivas i `Uppgift4.m`. Skriv ut lösningen (värdet på w) till Command Window. (1p)

(c) I mappen `Uppgift4` finns en funktionsfil `largestOfThree.m` som beskriver en funktion som tar emot tre inargument och svarar med att returnera det inargument som var störst (om man t ex evaluerar *largestOfThree(1,2,3)* ska svaret bli 3). Funktionen innehåller dock i nuläget två fel! Hitta dessa fel och rätta dem. Svara med en rättad `largestOfThree.m`. (Notera att du ska hitta och rätta felen, inte skriva en helt ny funktion som löser problemet på ett annat sätt). (2p)

Om någonting händer med `largestOfThree.m` kan du kopiera orginalet från underkatalogen `backup_kopia`.

5. 132-mönster (9p)

Ett så kallat 132-mönster är ett talmönster som kännetecknas av att det, i en vektor *nums* med n heltal, finns tre heltal *nums(i)*, *nums(j)* och *nums(k)* sådana att $i < j < k$ och $nums(i) < nums(k) < nums(j)$.

Skriv en funktion `find132Pattern.m` som tar emot en vektor med heltal och svarar sant (**true**) om det finns ett 132-mönster i denna vektor, annars falskt (**false**). Spara funktionen i mappen `Uppgift5`. Testa din funktion med fler olika inargument av olika längd för att säkerställa att den fungerar som avsett. Skriv slutligen ett skript `Uppgift5.m` som anropar `find132Pattern` med en valfri testvektor.

```
>> find132Pattern([1 2 3 4])
ans =
  logical
   0
>> find132Pattern([3 1 4 2])
ans =
  logical
   1
>> find132Pattern([-1 3 2 0])
ans =
  logical
   1
```

Figur 4: Illustration av hur det kan se ut när man anropar `find132Pattern` från Command Window med olika vektorer. `[1 2 3 4]` har inget 132-mönster, medan `[3 1 4 2]` har ett (`[1 4 2]`). `[-1 3 2 0]` har tre olika 132-mönster (`[-1 3 2]`, `[-1 3 0]` och `[-1 2 0]`).