

2 ukers HJEMMEEKSAMEN

PG3401 C Programmering

Tillatte hjelpemidler: Alle

Varighet: 14 dager

Karakterskala/vurderingsform: Nasjonal karakterskala A - F

Dato: 27. april - 11. mai 2023

This is the exam text in Norwegian, see PG3401-Homeexam-14day-V23-EN.pdf for the English version of this text. All tasks in the exam set are the same in English and Norwegian versions, only language differs. You can hand in your exam in either English or Norwegian (Swedish and Danish is also accepted).

Oppgavesettet har 7 sider. Det er totalt 6 oppgaver i oppgavesettet.

Det er 2 ukers frist på denne hjemmeeksamen. Perioden kan overlappe med andre eksamener dere har, det er derfor viktig at dere bruker tiden effektivt og planlegger tiden godt, så dere ikke rett før innlevering blir sittende og skulle levere flere eksamener samtidig. Vær obs på at eksamen MÅ leveres innen fristen som er satt i Wiseflow, og oppgaven kan kun leveres via WISEFLOW. Det vil ikke være mulig å få levert oppgaven etter fristen – det betyr at du bør levere i god tid slik at du kan ta kontakt med eksamenskontoret eller brukerstøtte hvis du har tekniske problemer.

Det presiseres at studenten skal besvare eksamen selvstendig og individuelt, samarbeid mellom studenter og plagiat er ikke tillatt. Det er ikke tillatt å presentere andres arbeid som ditt eget – dette inkluderer arbeid utført av kunstig intelligens (som tekst- eller kode-genereringsmodeller). Eksamen skal løses på Linux.

Merk at oppgavene er laget med stigende vanskelighetsgrad, og spesielt de tre siste oppgavene er vanskeligere enn de første oppgavene. Det oppfordres derfor til å gjøre de første oppgavene (helt) ferdige slik at studenten ikke bruker opp all tid på å gjøre de siste oppgavene først.

Format på innlevering

Dette er en praktisk programmeringseksamen (bortsett fra oppgave 1), fokus bør derfor være på å forklare hvordan du har gått frem, begrunne valg og legge frem eventuelle antagelser du har gjort i din løsning.

Hvis du ikke klarer å løse en oppgave er det bedre om du leverer inn det du har gjort (selv om det ikke virker), og forklarer hvordan du har gått frem og hva du ikke fikk til – enn å ikke besvare oppgaven i det hele tatt. Det forventes at alt virker hvis ikke annet er beskrevet i PDF filen, kode som studenten vet ikke virker bør også være kommentert i koden. Hvis du vet at programmet krasjer, ikke kompilerer eller ikke

virket slik den var tenkt er det viktig å forklare dette sammen med hvilke skritt du har tatt for å forsøke å løse problemet.

Besvarelsen skal være i 1 ZIP fil, navnet på filen skal være PG3401_V23_[kandidatnummer].zip. Denne filen skal ha følgende struktur:

```
\ task_2 \ makefile  
\ task_2 \ [...]  
[...]
```

I Wiseflow skal du laste opp en tekstbesvarelse med navn «PG3401_V23_[kandidatnummer].pdf», og ZIP filen skal lastes opp som vedlegg til tekstbesvarelsen.

Vær sikker på at alle filer er med i ZIP filen. Hver mappe skal ha en makefile fil, og det skal ikke være nødvendig med noen endringer, tredjeparts komponenter eller parametere – sensor vil i shell på Debian Linux 10 gå inn i mappen og skrive «make» og dette skal bygge programmet med GCC.

Tekstbesvarelsen skal inneholde besvarelse på oppgave 1 (skriv det kort og konsist, trenger ikke noe stor avhandling). Etter den rene tekstbesvarelsen skal det være 1 sides begrunnelse/dokumentasjon for hver oppgave, hver av disse begrunnelsene skal være på en ny side for å gjøre tekstbesvarelsen oversiktlig for sensor.

Besvarelsen skal være i PDF format og ha korrekt file-extension til å kunne åpnes på både en Linux og en Windows maskin (.pdf). Besvarelser i andre formater vil ikke bli lest.

Oppgave 1. Teori (5 %)

- a) Forklar hva C programmeringsspråket kan brukes til.
- b) Hvem er Dennis Ritchie og hva er han kjent for innen Informasjonsteknologi?
- c) Forklar hva kommandoen sudo gjør i terminal på Linux, og hva man typisk bruker denne kommandoen til ved bruk / administrasjon av Linux.

Oppgave 2. Filhåndtering og funksjoner (15 %)

Last ned følgende inn-data fil, den inneholder 10 heltall (integer) hvor hvert tall står på en linje (det er linjeskift mellom hvert tall):

http://www.eastwill.no/pg3401/eksamen_v23_oppgave2.txt

Last så ned følgende kildefiler:

http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_fib.c

http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_prim.c

http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_kvad.c

http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_cube.c
http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_perf.c
http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_abun.c
http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_def.c
http://www.eastwill.no/pg3401/eksamen_v23_oppgave2_odd.c

Du skal skrive et program som leser tallene i tekstfilen, for hver av tallene i tekstfilen skal du kalle funksjonene i de overnevnte kildefilene for å finne ut om tallene er a) et tall i Fibonacci-rekken, b) et primtall, c) et kvadrattall, d) et kubisk tall, e) et perfekt tall, f) et overflødig tall, g) et defekt tall, eller h) et oddetall.

Programmet skal opprette en output fil hvor hvert tall skrives ut BINÆRT, sammen med metadata om tallet i henhold til resultat fra funksjonene oppgitt over. Hvert tall i output filen skal beskrives med følgende struct:

```
struct OPPGAVE2_TALL_METADATA {  
    int iIndex; // Rekkefølge i filen, første tall = 1  
    int iNumber; // Tallet, som lest fra input filen  
    bool bIsFibonacci;  
    bool bIsPrimeNumber;  
    bool bIsSquareNumber;  
    bool bIsCubeNumber;  
    bool bIsPerfectNumber;  
    bool bIsAbundantNumber;  
    bool bIsDeficientNumber;  
    bool bIsOddNumber;  
}
```

Du må lage en makefile fil (bruk makefile fil hentet fra forelesning 4, sleide 65 – med overskriften «Use this makefile») som linker sammen kildefilene, og en eller flere header filer for programmet til å kompilere riktig, gjør eventuelle endringer påkrevet i makefile filen og kildefilene for å få programmet til å bygge og kjøre korrekt.

Både input filen og output filen skal være i samme katalog som programmet, og begge filene skal være en del av innlevert besvarelse.

Oppgave 3. Liste håndtering (20 %)

Du skal lage en enkel data-struktur for å håndtere flyavganger. Listens «stamme» skal være en dobbeltlenket liste over fly, hvert element (struct) i listen skal inneholde pekere til både forrige element og neste element. Elementet skal også inneholde en tekststreng som er FLYID (for eksempel BA-42), en tekststreng som beskriver DESTINASJON, en integer for antall SETER og en integer som skal fungere som TID for avgang. Hvert element skal i tillegg inneholde en enkeltlenket liste over (reservasjoner for) PASSASJERER.

Hvert element i passasjer listen (reservasjonene) skal inneholde en integer som inneholder SETENUMMER, en tekststreng med NAVN, og en integer som inneholder ALDER på den reisende. Listen skal alltid være SORTERT etter setenummer.

Du skal lage funksjoner som utfører følgende operasjoner på listen:

- Legge til en flyavgang i listen
- Legge til en passasjer til en flyavgang (husk at listen til enhver tid skal være sortert etter setenummer)
- Henter ut en flyavgang N fra listen (telt fra starten av listen, hvor første element er element nummer 1) og skrive ut på skjermen alle dataene tilhørende avgangen inklusive liste over passasjerene
- Finner flyavgang som matcher avreise TID i listen, og returnerer hvilket «element nummer» den har – ref funksjonen over
- Sletter en flyavgang (og alle passasjers reservasjoner for avgangen)
- Sletter en passasjers reservasjon (på en flyavgang)
- Bytter sete for en passasjer
- Søke gjennom listene etter en passasjers NAVN (i alle flyavgangene) og returnere ANTALL flyavganger denne passasjeren er forbundet med

Du skal lage en main funksjon som mottar instruksjoner fra bruker basert på input fra tastaturet, main må altså kunne kalle alle de åtte funksjonene over (for eksempel i en form for meny) og be brukeren om data som er nødvendig for å kalle de nevnte funksjoner. Main skal rydde opp alle data før den returnerer (et valg i menyen må være å avslutte).

Oppgave 4. Tråder (15 %)

I praktisk programmering er det ofte effektivt å legge tidkrevende operasjoner ut i arbeidstråder, eksempler på dette er filoperasjoner, nettverksoperasjoner og kommunikasjon med eksterne enheter. I denne oppgaven skal du simulere slike operasjoner med et mindre datasett – for å spare dere for tid under eksamen er det i denne oppgaven allerede laget en ferdig applikasjon som dere skal endre.

Applikasjon består av 3 tråder, hovedtråden (som kjører main funksjonen) og 2 arbeidstråder. Hovedtråden starter arbeidstrådene med mekanismer for tråd-kommunikasjon og synkronisering (i denne applikasjonen har ikke hovedtråden noen

annen funksjon, og starter kun de to trådene som gjør selve jobben). Trådene har et minneområde med plass til 4096 byte for kommunikasjon mellom trådene.

Den ene arbeidstråden (tråd A) skal lese en tekstfil, arbeidstråd A skal så sende filen over til den andre arbeidstråden (tråd B) gjennom flere sykluser ved hjelp av minneområdet beskrevet over, og signalere tråd B om at det er data tilgjengelig i bufferet. Tråd B teller så opp antall forekomster av bytes med verdi 00, 01, 02, og så videre til; FF i filen den får sendt over. Tråd A og tråd B går i loop for å prosessere filen helt til den er ferdig. Når filen er sendt over i sin helhet avslutter arbeidstråd A. Arbeidstråd B fullfører sin opptelling av bytes, og så skriver den ut til terminal vinduet antall forekomster av hver av de 256 mulige byte-verdiene, før den også avslutter. Hovedtråden (main) venter på at begge trådene avslutter, rydder opp korrekt og så avslutter applikasjonen.

Last ned følgende kildefil, dette er en løsning på applikasjonen som beskrevet:

http://www.eastwill.no/pg3401/eksamen_v23_oppgave4.c

Last ned en test-fil som kan brukes til denne oppgaven (Hamlet av Shakespeare, hentet fra Project Gutenberg - <https://www.gutenberg.org/ebooks/2265>):

http://www.eastwill.no/pg3401/eksamen_v23_oppgave4_pg2265.txt

Du skal utvide koden med følgende endringer:

- Du må lage en makefile fil (bruk makefile fil hentet fra forelesning 4, sleide 65 – med overskriften «Use this makefile») for å programmet til å bygge, gjør eventuelle endringer påkrevet i makefile filen og kildefilen for å få programmet til å bygge og kjøre korrekt
- Programmet bruker globale variable, endre dette til at alle variable er lokale variable i main funksjonen og sendes inn som parameter til de to trådene
- Tråd A har hardkodet navnet på filen som skal leses, endre programmet til å ta filnavnet som parameter på kommandolinjen og sender dette videre inn til Tråd A som en variabel
- I stedet for å telle «byte verdier» (fra 0 til 255) skal du utvide funksjonaliteten til å telle bokstaver og printbare spesialtegn (ascii kode 32 til 126), du skal også endre koden som skriver ut antall forekomster på skjerm til å angi hvilken bokstav / tegn dette er – for eksempel 'A' : ?
- I tillegg skal du telle forekomsten av et par vanlige korte ord; «and», «at», «it», «my», samt navnet «Hamlet», antall forekomster skal skrives ut på skjerm etter at bokstavene i forrige punkt er skrevet ut
- Skriv om koden fra å bruke conditions til å bruke semaforer
- Legg til kode for eksplisitt initialisering av mutex og semaforer (med *_init funksjonene)
- Legg til kommentarer i koden for å dokumentere hva koden gjør

Oppgave 5. Nettverk (25 %)

Du skal i denne oppgaven lage et verktøy for å sette opp et reverse shell på en maskin. Et reverse shell er en applikasjon hvor en tjener sender kommandoer til en klient som klienten kjører «i terminal» som om et menneske hadde sittet på klienten og manuelt skrevet de samme kommandoene i terminal vinduet. Applikasjonen skal kunne startes både som klient og som server, og du skal lage din egen protokoll for kommunikasjon mellom klient og server. Når startet som server (for eksempel signalert ved et parameter fra terminal, som kan være -listen) skal applikasjonen åpne oppgitt port for LISTEN, for denne oppgaven holder det å binde til loopback på 127.0.0.1 (for å ikke eksponere en åpen port eksternt). Det anbefales å bruke TCP.

Oppgaven skal ikke løses ved bruk av Curl eller andre tredjepartsbiblioteker, og skal ikke basere seg på å starte andre programmer i operativsystemet – kun bruk av Sockets slik vi har lært på forelesning 10 om Nettverk vil gi poeng på oppgaven.

Når startet som server skal den BINDE til en port brukeren velger, det anbefales for denne oppgaven å kun lytte på adresse 127.0.0.1 for å ikke eksponere porten utenfor egen maskin. Det anbefales å bruke TCP. Server applikasjonen skal eksekveres med portnummer som parameter fra terminal, for eksempel «oppgave_5 -listen -port 42».

Når startet som klient skal applikasjonen eksekveres med serverens IP adresse og portnummer fra terminal, for eksempel «oppgave_5 -server 127.0.0.1 -port 42». Når den starter skal applikasjonen CONNECTE til oppgitt port på server-prosessen.

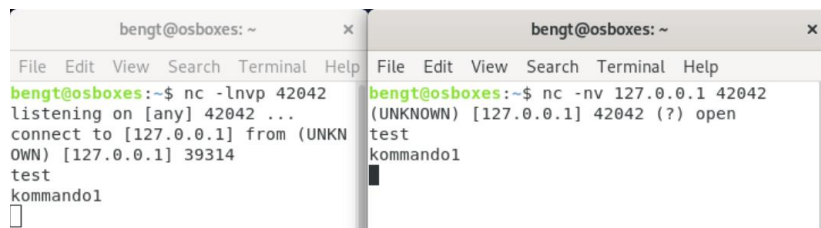
Du skal lage en protokoll for trafikk mellom server og klient, som et minimum må denne inneholde KOMMANDO som en tekststreng som sendes til klienten, og RESULTAT som en tekststreng som sendes tilbake til serveren. Når en klient er tilkoblet skal server applikasjonen akseptere input fra brukeren i form av tekststrenger i terminal, disse skal være Linux (terminal) kommandoer som skal sendes til klient applikasjonen på et format studenten velger. Klient applikasjonen skal EKSEKVERE disse kommandoene, og sende RESULTATET tilbake til serveren som printer dette ut på skjerm. Eksempel på kode som gjør dette (som dere kan bruke i besvarelsen) kan dere finne her:

http://www.eastwill.no/pg3401/eksamen_v23_oppgave5_exec.c

Begge applikasjoner skal kunne avsluttes med Ctrl-C, begge applikasjoner bør håndtere dette (uten å krasje). Når en applikasjon avslutter (enten klient eller server) skal den andre applikasjonen også avslutte.

Et tips for å teste server/klient applikasjoner er å åpne to terminal vinduer i Linux og starte den ene instansen «oppgave_5 -listen -port 42» i ett vindu og den andre instansen «oppgave_5 -server 127.0.0.1 -port 42» i det andre vinduet. Eksempel (fra verktøyet netcat som er et eksisterende verktøy som kan gjøre det samme som

deres oppgave skal gjøre):



The image shows two terminal windows side-by-side. The left window is titled 'bengt@osboxes: ~' and shows a Netcat listener on port 42042. It receives a connection from 127.0.0.1. The right window is also titled 'bengt@osboxes: ~' and shows a Netcat client connecting to 127.0.0.1 on port 42042. Both windows show the exchange of the words 'test' and 'kommando1'.

```
bengt@osboxes: ~  
File Edit View Search Terminal Help  
bengt@osboxes:~$ nc -lnvp 42042  
listening on [any] 42042 ...  
connect to [127.0.0.1] from (UNKN  
OWN) [127.0.0.1] 39314  
test  
kommando1  
█
```

```
bengt@osboxes:~$ nc -nv 127.0.0.1 42042  
(UNKNOWN) [127.0.0.1] 42042 (?) open  
test  
kommando1  
█
```

Oppgave 6. Filhåndtering og tekst-parsing (20 %)

Du skal lage en applikasjon som fungerer som en «kode beautifier», en applikasjon som endrer kildekode til noe som passer forfatterens kodelstil (gjør koden «penere»).

Applikasjonen skal ta et filnavn til en C-source fil som parameter når den startes fra terminal. Applikasjonen skal lese denne filen og gjøre 3 endringer i filen før den lagrer filen med samme navn, men lagt til `_beautified` før `.c` i filnavnet.

A) Først skal applikasjonen ta alle forekomster av while-looper og erstatte disse med for-looper, det betyr at kode som dette:

```
a = 0;  
while (a < b) {... a++; }
```

skal erstattes med en for loop og vil se noe slik ut:

```
for (a = 0; a < b; a++) {...}
```

Hvor ... angir resten av koden i løkken, og naturligvis må beholdes slik den er. Det kan forutsettes at kun «enkle» løkker (som burde vært en for-loop) hvor en variabel settes til en verdi rett før løkken og løkken har en enkel test med denne verdien (som i eksempelet over) erstattes – og alle andre bruk av while beholdes slik de er.

B) Applikasjonen skal så tvinge frem bruk av Hungarian notation i koden, dette skal løses ved å finne alle variable av typen «unsigned int» og endre navn på alle variable av den typen til å være «uiAbc» hvor Abc er variabelens opprinnelige navn, eksempel vil «unsigned int counter» resultere i at alle forekomster av variabelen endres til «uiCounter». (Du skal ikke gjøre dette for andre typer, da det blir for mye jobb.)

C) I tillegg skal alle forekomster av 3 mellomrom (space) erstattes med en tabulator (ASCII kode 0x09).

+

Slutt på oppgavesettet