2 ukers HJEMMEEKSAMEN
PG3401 C Programmering
Tillatte hjelpemidler: Alle
Varighet: 16 dager
Karakterskala/vurderingsform: Nasjonal karakterskala A - F
Dato: 3.-19. november 2020

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

Det er 2 ukers frist på denne hjemmeeksamen, men forventet arbeidsmengde er 5-7 dager med «normale» arbeidsdager. Perioden kan overlappe med andre eksamener dere har, det er derfor viktig at dere bruker tiden effektivt i starten av eksamensperioden 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.

Merk at oppgavene er laget med stigende vanskelighetsgrad, og spesielt de to 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.

Spørsmål fra studenter i løpet av eksamensperioden skal stilles under Diskusjoner på emnesidene på Canvas; svar vil kun gis der slik at alle studenter får den samme informasjonen. Det betyr at alle studenter bør følge med på Canvas under Diskusjoner for å få med seg eventuelle presiseringer og oppklaringer som måtte komme som en følge av spørsmål fra medstudenter.

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 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 tekstbesvarelsen, 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_H20_[kandidatnummer]. Denne filen skal ha følgende struktur:

\ Tekstbesvarelse_PG3401_H20_[kandidatnummer].pdf
\ oppgave_2 \ makefile
\ oppgave_2 \ […]
[…]

Vær sikker på at alle filer er med i ZIP filen. Hvis du velger å ha flere programmer som en del av løsningen (for eksempel oppgave 7) så legger du det ene programmet i \oppgave_7A og den andre i \oppgave_7B. 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 eller i Microsoft Word format (DOCX) og ha korrekt file-extension til å kunne åpnes på både en Linux og en Windows maskin (.pdf eller .docx). Besvarelser i andre formater vil ikke bli lest.

Oppgave 1. Generelt (5 %)

a) Forklar hva C programmeringsspråket kan brukes til.

b) Hvem er Linus Torvalds og hva er han kjent for innen Informasjonsteknologi?

c) Forklar hva kommandoen **sudo** gjør i Linux shell, og hvorfor det brukes (ofte).

Oppgave 2. Sortering (15 %)

Du skal lage en funksjon for å sortere en tabell (array) av struct variable med metoden Bubble Sort. Du skal så lage et program hvor main metoden tar som parameter en rekke med tekststenger, oppretter en tabell som populeres med input data og sender denne tabellen til sorteringsfunksjonen. Etter at dataene er sortert skal main funksjonen skrive ut alle strengene i sortert rekkefølge i konsoll vinduet (for eksempel med printf), med et streng per linje.

Struct variabelen skal inneholde tekststrengen, en int som sier hvor mange bokstaver som er i strengen, og en peker til arrayet structen er en del av.

I tillegg til main og sorteringsfunksjonen skal du også lage en funksjon som tar et enkelt element (en struct) som parameter – og returnerer hvilken index inn i tabellen dette elementet ligger på.

Sorteringsfunksjonen skal ta hele tabellen av structer som parameter og sorterer de alfabetisk slik at når funksjonen returnerer er hele tabellen sortert.

Oppgave 3. Liste håndtering (15 %)

Du skal lage en dobbeltlenket liste, hvert element (struct) i listen skal inneholde pekere til både forrige element og neste element. Elementet skal også inneholde en tekststreng som inneholder NAVN, en integer som inneholder ALDER, og en tekststreng som inneholder KOMMUNE vedkommende bor i.

Du skal lage funksjoner som utfører følgende operasjoner på listen:

- Legge til et element i listen
- Henter ut element N fra listen (telt fra starten av listen, hvor første element er element nummer 1) og returnerer alle dataene i elementet
- Finner element som matcher NAVN i listen, og returnerer hvilket «element nummer» den har – ref funksjonen over
- Sletter alle elementer i listen som har et gitt NAVN (sendes som parameter til funksjonen)
- Sletter alle elementer i listen som har ALDER lavere eller høyere eller lik en gitt verdi (sendes som parameter til funksjonen)

Du skal lage en main funksjon som mottar instrukser fra bruker basert på input fra tastaturet, main må altså kunne kalle alle de fem 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. Finn 3 feil (10 %)

Du har fått beskjed om å utføre feilretting av denne funksjonen, du får beskjed om at den krasjer innimellom. Support har kommet frem til at hvis kortnummer tilhører bankens nummerserie (4242 123x) og serien slutter på 9 så krasjer betalingen (med andre ord hvis det er tallet 9 som det 8ende tallet i kortnummeret) hos mange brukere.

Det viser seg at koden faktisk inneholder 3 feil. Du skal rette alle feilene, i besvarelsen skal du ha den feilrettede funksjonen, og en main metode som tester funksjonen (der hvor det er mulig bør main kalle funksjonen på en slik måte at den fremprovoserer feilene. I tekstbesvarelsen skal du kort forklare hvilke feil du fant, hvorfor dette er feil og forklare hvordan du løste dem.

I tillegg mangler det kode for å sjekke om kortet er utløpt, du skal skrive denne koden og legge den til i funksjonen. (Parameter e til funksjonen er utløpsdato xx/xx…)

OBS: Ikke inkluder noe testkode med dine egne kredittkort nummer i besvarelsen, Wiseflow er ikke PSI DSS godkjent :-)

```
typedef struct _CCD { char digit[5]; int convert; struct _CCD* p; } CCD;
int Backendprocesspayment(int a, int64_t c, CCD* p, const char* n, int b);

int ProcessCreditcardPayment(int a, char* c, char* e, char* n) {
   int64_t llCreditcard = 0; /* C99 type, our compiler supports this... */
   char *i = (char*)c; int j = 0;
   CCD *pc, *cc = (CCD*)malloc(sizeof(CCD));
   if (!cc) exit(1); else pc = cc;
   memset(cc, 0, sizeof(CCD));
   /* Create 4 linked structures that holds one 4 digit
      segment of cardnumber: */
   while (i[0]) {
      pc->digit[j++] = i++[0];
      if (strlen(pc->digit) == 4) {
         pc->p = (CCD*)malloc(sizeof(CCD));
         if (!pc->p) exit(1);
         else {memset(pc->p, 0, sizeof(CCD)); pc = pc->p; j=0;}
      }
   }
   /* Check that card starts with 4242, if not card is from
      another bank so we fail: */
   if (strcmp(cc->digit, "4242") != 0) { free(cc); return 6; }
   /* Calculate the cardnumber as a 64 bit integer: */
   for (j = 12, pc = cc; pc; pc = pc->p, j-=4) {
      pc->convert = atoi(pc->digit);
      llCreditcard += ((int64_t)pc->convert) * pow(10, j);
   }
   /* If next section is 123x it is a bonus card with cash-back, we
      send type (x) to backend below: Set j to the type of bonus card */
   if (strncmp(cc->p->digit, "123", 3) == 0) {
      j = cc->p->digit[cc->p->digit[3]/((cc->p->digit[3]-'0')%9)];
   }
   /* Call backend function: */
   Backendprocesspayment(a, llCreditcard, cc, n, j);

   free(cc);
   return 0;
}
```

Oppgave 5. Filhåndtering (15 %)

Opprett en fil med følgende innhold:


```
QXJiZWlkZXQgbWVkIGRldHRlIGVtbmV0IHNrYWwgZ2kgc3R1ZGVudGVuIGlubm
ZyaW5nIGkgcHJvZ3JhbW1lcmluZ3NzcHJrZXQgQyBvZyBodm9yZGFuIGRldHRl
IGthbiBicnVrZXMgdGlsICBpbnRlcmFnZXJlIGRpcmVrdGUgbWVkIG9wZXJhdG
l2c3lzdGVtZXQuIERldCBza2FsIG9ncyBnaSBmb3JzdGVsc2UgYXYgb3BlcmF0
aXZzeXN0ZW1ldCBMaW51eCBvZyBrdW5uc2thcCBvbSBodm9yZGFuIGRldHRlIH
Zpcmtlci4=
```

Teksten er en tekst som er enkodet med BASE64. Både den enkodede og dekodede teksten er i 7 bit ASCII, og den enkodede teksten er på 1 linje (se bort fra linjeskift du ser i denne oppgaveteksten). Skriv et program som leser denne filen, dekoder BASE64 strengen, og skriver resultatet til en annen fil.

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

(Obs; input teksten er norsk tekst, men norske spesialtegn er borte fra teksten for å få den 7-bit ASCII kompatibel. Output teksten etter å ha løst oppgaven kan derfor se «feil» ut, men fortsatt være riktig.)

Oppgave 6. Tråder (20 %)

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.

Du skal lage en applikasjon som består av 2 tråder, hovedtråden (som startet main) og en arbeidstråd som skal lese data. Applikasjonen skal ta navnet på en TEKST fil bestående av 7-bit ASCII som input parameter på kommandolinje, etter at applikasjonen har startet skal den starte arbeidstråden med mekanismer for tråd-kommunikasjon og synkronisering. Hovedtråden og arbeidstråden skal ha et minneområde med plass til 10 tegn. (Det kan være en struct som også inneholder andre kontrolldata som for eksempel antall tegn i bufferet og annet studenten finner nyttig – men ikke mer enn 10 tegn fra tekstfilen som leses. Det er lov for en slik struct å inneholde char a[11] hvis det ellevte tegnet kun brukes til en 0-terminering.)

Arbeidstråden skal lese 10 tegn fra filen og kopiere til det delte minneområdet, og så signalere hovedtråden at det er data tilgjengelig. Hovedtråden skal når det er data i minneområdet lese dataene og skrive ut teksten i konsoll vinduet (for eksempel med printf), når data er lest ut fra bufferet skal data merkes som lest (for eksempel ved å sette dataene til 0) og så skal den sende et signal til arbeidstråden at den kan fortsette å lese data.

Slik skal hovedtråden og arbeidstråden kjøre vekselsvis til hele filen er lest, arbeidstråden skal da avslutte automatisk. Når hovedtråden er ferdig skal hele applikasjonen avslutte – før hovedtråden avslutter skal den skrive ut totalt antall tegn som har blitt lest (og skrevet ut) fra filen.

Sensor vil for eksempel kunne teste det kompilerte programmet med en av kildefilene til programmet, som typisk vil være en ASCII tekstfil.

```
> ./oppgave_6 oppgave_6.c
```

Oppgave 7. Nettverk (20 %)

Du skal lage 1 server applikasjon og 1 klient applikasjon. Et tips for å teste denne oppgaven under programmeringen er å åpne 2 terminalvinduer, og kjøre hver applikasjon i eget vindu.

Server applikasjonen skal eksekveres med portnummer og en brukerspesifisert ID på kommandolinjen, for eksempel «oppgave_7a -port 42 -id Meningen_med_livet». Når startet skal applikasjonen åpne oppgitt port for LISTEN, for denne oppgaven holder det å binde til loopback på 127.0.0.1. Det anbefales å bruke TCP.

Klient applikasjonen skal eksekveres med serverens portnummer på kommandolinje, for eksempel «oppgave_7b -server 42». Når startet skal applikasjonen CONNECTE til oppgitt port på serverapplikasjonen.

Det skal være en form for handshake mellom klient og server hvor som et minimum server sender ID som ble spesifisert på kommandolinje, og når klient mottar denne IDen skal applikasjonen spørre brukeren «Forsøker å koble til server Meningen_med_livet, ønsker du å fortsette Y/N» og vente på brukerens input. Hvis brukeren svarer N skal handshake avsluttes og klient applikasjonen avslutte, svarer brukeren Y skal handshake fullføres.

Klient applikasjonen skal akseptere input fra brukeren i form av tekststrenger på kommandolinje, disse tekststrengene skal sendes til server applikasjonen på et format studenten velger. Server applikasjonen skal skrive ut mottatte tekststrenger i konsoll vinduet (for eksempel med printf).

Begge applikasjoner skal kunne avsluttes med Ctrl-C, begge applikasjoner bør håndtere dette (uten å krasje).

+

Slutt på oppgavesettet.