

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1001 **– Grunnleggende programmering**

Faglig kontakt under eksamen: Frode Haug

Tlf: 950 55 636

Eksamensdato: 26.november 2019

Eksamenstid (fra-til): 09:00-13:00 (4 timer)

Hjelpemiddelkode/Tillatte hjelpemidler: F - Alle trykte og skrevne.
(kalkulator er *ikke* tillatt)

Annen informasjon:

Målform/språk: Bokmål

Antall sider (inkl. forside): 7

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig ☒ 2-sidig ☐

sort/hvit ☒ farger ☐

Skal ha flervalgskjema ☐

Kontrollert av:

Dato

Sign

NB: Oppgave 1a, 1b og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30%)

a) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <stdio.h>
#include <string.h>

char txt[] = "DETTE-ER-DEN-ALLER-ALLER-FORSTE-EKSAMEN-I-DETTE-NYE-EMNET-PAA-NTNU";

int main() {
    int i = 41 / 3,    j = 1 + 6 * 3,    k = 0,    n = strlen(txt);

    while (txt[i++] == txt[j]) printf("%c", txt[j++]);
    printf("\n");

    printf("%c %c %c\n", txt[i], txt[i + 3], txt[j]);

    for (i = 1; i <= n; i++)
        if (txt[i - 1] == 'E') k++;
    printf("%i\n", k);

    i = n % 8;    j = (5 * 6) % 9;    k = i * j;
    printf("%c %c %c\n", txt[i - 1], txt[j - 1], txt[k + 3]);

    while (txt[j] != txt[k]) {
        printf(" %c", txt[j]);    j += i;    k += i;    j++;
    }
    printf("\n");

    return 0;
}
```

b) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

struct H19 { char txt1[10], txt2[30]; int ant; };

void H19Funk1(const struct H19 h) {
    printf("%s - %s, %ix", h.txt1, h.txt2, h.ant); }

int H19Funk2(const struct H19* h1, const struct H19* h2) {
    return ((h1->ant < h2->ant) ? h2->ant : h1->ant); }

bool H19Funk3(const struct H19 h1, const struct H19 h2) {
    return (h1.txt1[4] == h2.txt1[4]); }

void H19Funk4(struct H19* h1, const struct H19* h2, const struct H19* h3) {
    strcpy(h1->txt1, h2->txt1); strcpy(h1->txt2, h3->txt2); h1->ant = h3->ant - 100; }

void H19Funk5(struct H19* h, const char* t) { strcat(h->txt2, t); }

int main() {
    struct H19 emne1 = {"PROG1001", "GrProg", 150},
        emne2 = {"IMT1031", "GrProg", 120}, emne3 = {"IMT2021", "AlgMet", 220};
    H19Funk1(emne1);    printf(" ");    H19Funk1(emne2);    printf("\n");
    emne1.ant = H19Funk2(&emne3, &emne2);    H19Funk1(emne1);    printf("\n");
    printf("%i\n", H19Funk3(emne2, emne3));
    H19Funk4(&emne1, &emne3, &emne2);    H19Funk1(emne1);    printf("\n");
    H19Funk5(&emne2, " - utgaar herved");    H19Funk1(emne2);    printf("\n");
    return 0;
}
```

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2g) *nøy*e, før du begynner å besvare noe som helst. Studér vedlegget (på utdelte papirer), som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `#define/const`, structen med datamedlemmer, funksjoner, globale variable (spesielt `int`'ene), `main()` og ferdiglagde funksjoner. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på ansatte på en arbeidsplass, hvilke dager de skal/ikke skal arbeide, og om de har møtt opp eller ei til arbeidet.

Datastrukturen

Datastrukturen består (se utdelt vedlegg) av arrayen `gAnsatte`. I denne er indeksene fra 0 til `gAntallAnsatte-1` i bruk. Vedlegget angir også hvilke datamedlemmer `Ansatt` inneholder. `jobbing` angir den ansattes jobbstatus for inneværende måned. Vi bruker indeksene 0 til *maksimalt* 30. Men det er variabelen `gSisteDag` som angir *eksakt* hva den siste dagen for inneværende måned er. Denne verdien blir satt ved at den leses inn fra fil helt i starten av programmet (se oppgave 2e). `gDagNr` er nåværende dag i måneden. Denne leses også inn først i programmet, og vi forutsetter at brukeren *alltid* skriver inn den *korrekte* dagen (en høyere enn den da programmet ble startet dagen før). Vedlegget inneholder *alt* du trenger av *stucter*, *datamedlemmer* og *globale variable* for å løse denne eksamensoppgaven. Dessuten er **prototyper** for funksjonene stort sett også deklarerert/definert.

Oppgaven

- a) **Skriv innmaten til** `void nyAnsatt()` **og** `void ansattLesOgSettData(.....)`
Den første funksjonen kommer med en melding om det ikke er plass til flere ansatte. I motsatt fall skrives den nye ansattes nummer ut på skjermen. **NB:** De ansatte har nr.1 - MAXANSATTE, selv om de ligger i element nr.0 - MAXANSATTE-1. Det allokeres så memory-plass til den nye ansatte. Vha. den andre funksjonen leses den nye ansattes tre første datamedlemmer. `jobbing` fylles med '-' for elementene 0 - `gDagNr-2`. Mens for intervallet `gDagNr-1` til `gSisteDag-1` leses det hvordan vedkommende *skal* jobbe *resten av denne måneden* (dvs. tegnene '-' eller 'J'). **NB:** Du trenger *ikke* å sjekke at brukeren *virkelig* skriver '-' eller 'J'.
- b) **Skriv innmaten til** `void ansattAnkommerJobb(.....)`
Funksjonen opererer *alltid* på element nr. `gDagNr-1` i `jobbing`. *Skal* den ansatte arbeide i dag (verdien 'J'), så settes status til møtt ('M'), og det kommer en liten melding om at dette er registrert. Det kommer også egne meldinger om: 1) vedkommende *ikke* skal arbeide i dag ('-'), 2) har allerede møtt ('M') eller 3) at det ligger et annet ulovlig/uaktuelt tegn i elementet. **NB:** Legg merke til hvordan funksjoner ifm. oppgave 2b og 2d startes opp av allerede ferdig laget kode i vedlegget.
- c) **Skriv innmaten til** `void oversiktAnsattesJobbing()` **og** `void ansattSkrivJobbing(.....)`
Den første funksjonen skriver først *en* linje med *alle* tallene fra 1 til `gSisteDag` (tre plasser til hvert tall). Deretter går den gjennom *alle* aktuelle ansatte. For *hver* av dem (vha. den andre funksjonen) skrives *hele* den ansattes jobbeplan for inneværende måned på *en* linje. Dvs. tegnene '-', 'J' og 'M' skrives med *to* blanke mellom hver, og til slutt på linjen skrives navnet.

- d) Skriv innmaten til** `void ansattEndreJobbing (.....)`
 Funksjonen leser først inn et aktuelt dagnummer f.o.m. den nåværende t.o.m. så mange dager det er i den aktuelle måneden. Denne dagens jobbstatus skrives så ut på skjermen. Brukeren må så skrive inn en *lovlig* verdi for jobbing (dvs. det *skal* sikres at verdien er en av de tre lovlig/aktuelle). *Inneværende dag* kan endres til hvilken som helst av de tre verdiene. *Fremtidige dager* kan *kun* endres til ‘-’ eller ‘J’ (selvsagt ikke mulig å ha møtt (‘M’) i fremtiden allerede), ellers kommer det en melding.
- e) Skriv innmaten til** `void lesJobbingFraFil (.....)` **og**
`void ansattLesJobbingFraFil (.....)`
 Filen «JOBGING.DTA» har følgende format:
 <N = aktuelt antall dager i inneværende måneden (i intervallet 28-31)>
 <Ansattnr> <Jobbstatus dag nr.1> <Jobbstatus dag nr.2> <Jobbstatus dag nr.N>
 Om `lesAlt` er false, leses *kun* N inn i `gSisteDag`. Ellers leses i tillegg *hele* resten av filens innhold inn til hver av de ansatte (vha. den andre funksjonen). Men, dette skjer *kun* etter at brukeren har svart ‘J’ til at det skal skje (siden det overskriver *alt* om jobbingen hos *alle* de ansatte). Derfor er kommandoen ‘M’ noe som *kun bør* utføres på morgenen den aller første dagen i måneden. Husk også å sjekke at ‘Ansattnr’ hele tiden er i lovlig intervall.
NB: `gAntallAnsatte` ligger *ikke* på fila. De ansatte/datastrukturen er allerede i memory.
- f) Skriv innmaten til** `void jobbStatistikk ()`
og all annen kode/funksjon(er) som trengs for å skrive ut hvor mange dager *hittil* i måneden *hver enkelt* ansatt har arbeidet/møtt og hvor mange dager vedkommende *ikke* har møtt.
- g) Skriv innmaten til** `void lesAnsatteFraFil ()` **og**
`void ansattLesFraFil (.....)`
 Funksjonene sørger til sammen for at *hele* datastrukturen ifm. de ansatte blir lest inn fra filen «ANSATTE.DTA». Om hver enkelt ligger det alltid data om 31 arbeidsdager, selv om noen av de bakerste for øyeblikket kanskje ikke er i bruk når programmet kjøres videre. Dette bestemmes jo av `gSisteDag`. **Filformatet** bestemmer du helt selv, men *skal oppgis i besvarelsen*.
NB: Husk å allokere memory-plass de stedene det er nødvendig.

Annet (klargjørende):

- Vi forutsetter at den som har ansvaret for å sette opp jobbplanen for neste måned husker å tilføye evt. nye ansatte sist måned på filen «JOBGING.DTA». Vedkommende orienterer også de ansatte (på en eller annen måte) tidlig nok om all fremtidig jobbing. For det blir jo for sent for ansatte å få vite om dette først når kommandoen ‘M’ utføres på den første dagen i en måned.
- Det skal altså *ikke* lages kode bl.a. for å: 1) skrive hele datastrukturen til fil, 2) skrive *alle* den ansattes data ut på skjermen.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av oppgaven.
- **NB:** Det skal *ikke* brukes C++-kode, dvs. slikt som f.eks: string-klassen, kode fra STL, templates eller andre større hjelpebiblioteker. Men, de vanligste inkluder brukt i hele høst er tilgjengelig.

Lykke til med all jobbing videre i livet!

FrodeH

Vedlegg til PROG1001, 26.nov. 2019: Halvferdig programkode

```
#include <stdio.h>           // printf, scanf, FILE
#include <stdlib.h>          // sizeof, malloc, free
#include <string.h>          // strcpy, strlen
#include <stdbool.h>         // bool, true, false
#include "LesData.h"         // Verktøykasse for lesing av diverse data

#define MAXDAG 31           ///< Max. antall dager i en måned.
#define MAXANSATTE 100      ///< Max. antall ansatte.
const int STRLEN = 80;      ///< Max. tekstlengde.

/**
 * Ansatt (med navn, mailadresse, mobiltlf og hvordan jobber DENNE måneden).
 */
struct Ansatt {
    char* navn,              // Ansattes navn.
    * mail;                  // Mailadresse.
    int mobil;               // Mobiltelefonnummer.
    char jobbing[MAXDAG];    // Jobbingen i løpet av måneden
};                           // ('-'= nei, 'J'= jobbe, 'M'= møtt)

void skrivMeny();
void nyAnsatt();             // Oppgave 2A
void ansattLesOgSettData(struct Ansatt* a); // Oppgave 2A
void kommerPaaJobb();        // (Oppgave 2B)
void ansattAnkommerJobb(struct Ansatt* a);   // Oppgave 2B
void oversiktAnsattesJobbing();             // Oppgave 2C
void ansattSkrivJobbing(const struct Ansatt* a); // Oppgave 2C
void endreEnsJobbingEnDag();                // (Oppgave 2D)
void ansattEndreJobbing(struct Ansatt* a);   // Oppgave 2D
void lesJobbingFraFil(const bool lesAlt);     // Oppgave 2E
void ansattLesJobbingFraFil(FILE* inn, struct Ansatt* a); // Oppgave 2E
void jobbStatistikk();                       // Oppgave 2F
void lesAnsatteFraFil();                     // Oppgave 2G
void ansattLesFraFil(FILE* inn, struct Ansatt* a); // Oppgave 2G

int gAntallAnsatte,          ///< Antall ansatte i bruk i 'gAnsatte'.
gDagNr,                     ///< Dagens dato.
gSisteDag;                  ///< Aktuell måneds dagantall (28-31).
struct Ansatt* gAnsatte[MAXANSATTE];        ///< Alle de ansatte.

/**
 * Hovedprogrammet:
 */
int main() {
    char kommando;

    lesAnsatteFraFil();       // Oppgave 2G
    lesJobbingFraFil(false);  // Oppgave 2E - Henter KUN 'gSisteDag'.
    gDagNr = lesInt("Dato", 1, gSisteDag); // Leser dagens dato (1-gSisteDag).

    skrivMeny();
    kommando = lesChar("Ønske");
    kommando = 'F';
    while (kommando != 'Q') {
        switch (kommando) {
            case 'N': nyAnsatt(); break; // Oppgave 2A
            case 'K': kommerPaaJobb(); break; // Oppgave 2B
            case 'O': oversiktAnsattesJobbing(); break; // Oppgave 2C
            case 'E': endreEnsJobbingEnDag(); break; // Oppgave 2D
            case 'M': lesJobbingFraFil(true); break; // Oppgave 2E
            case 'S': jobbStatistikk(); break; // Oppgave 2F
            default: skrivMeny(); break;
        }
        kommando = lesChar("Ønske");
    }
    // skrivAnsatteTilFil(); // Skriver ALLE ansatte tilbake til fil.
    printf("\n\n");
}
```

```

    return 0;
}

/**
 * Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tN   = Ny ansatt\n");
    printf("\tK   = ansatt Kommer på jobb\n");
    printf("\tO   = Oversikt over alle ansattes jobbing\n");
    printf("\tE   = Endre jobbing for en ansatt på en dag\n");
    printf("\tM   = Månedsskifte (ny jobbplan)\n");
    printf("\tS   = Statistikk over antall dager jobbet/ikke møtt\n");
    printf("\tQ   = Quit/avslutt\n");
}

/**
 * Oppgave 2A - Legger inn (om mulig) en ny ansatt i datastrukturen.
 *
 * @see   ansattLesOgSettData(...)
 */
void nyAnsatt() {                                     /*   LAG INNMATEN   */ }

/**
 * Oppgave 2A - Leser inn og initierer data om EN ansatt.
 *
 * @param a - Den ansatte som får innlest/initiert sine data
 */
void ansattLesOgSettData(struct Ansatt* a) {          /*   LAG INNMATEN   */ }

/**
 * Oppgave 2B - Ansatt har kommet på jobb.
 *
 * @see   ansattAnkommerJobb(...)
 */
void kommerPaaJobb() {
    int nr = lesInt("Ansattnr", 1, gAntallAnsatte); // Leser aktuelt ansnr.
    ansattAnkommerJobb(gAnsatte[nr - 1]);
}

/**
 * Oppgave 2B - Registrerer (om mulig) at en ansatt har møtt på jobb.
 *
 * @param a - Den ansatte som får registrert at har møtt på jobb
 */
void ansattAnkommerJobb(struct Ansatt* a) {           /*   LAG INNMATEN   */ }

/**
 * Oppgave 2C - Skriver ALLE de ansattes jobbstatus.
 *
 * @see   ansattSkrivJobbing(...)
 */
void oversiktAnsattesJobbing() {                     /*   LAG INNMATEN   */ }

/**
 * Oppgave 2C - Skriver EN ansatts navn og jobbstatus.
 *
 * @param a - Den ene ansatte som jobbstatus skrives for
 */
void ansattSkrivJobbing(const struct Ansatt* a) {     /*   LAG INNMATEN   */ }

```

```

/**
 * Oppgave 2D - Endre en ansatts jobbing. FERDIGLAGET
 *
 * @see ansattEndreJobbing(...)
 */
void endreEnsJobbingEnDag() {
    int nr = lesInt("Ansattnr", 1, gAntallAnsatte); // Leser aktuelt ansnr.
    ansattEndreJobbing(gAnsatte[nr - 1]);
}

/**
 * Oppgave 2D - Endrer (om mulig) status for jobbing.
 *
 * @param a - Den ansatte som får endret sin jobbing
 */
void ansattEndreJobbing(struct Ansatt* a) { /* LAG INNMATEN */ }

/**
 * Oppgave 2E - Jobbingen for NESTE måned leses inn i den
 * allerede eksisterende datastrukturen.
 *
 * @param lesAlt - Om HELE filens innhold skal leses, eller KUN 'gSisteDag'
 * @see ansattLesJobbingFraFil(...)
 */
void lesJobbingFraFil(const bool lesAlt) { /* LAG INNMATEN */ }

/**
 * Oppgave 2E - Leser KUN jobbstatus for EN ansatt fra fil.
 *
 * @param inn - Filen det skal leses fra
 * @param a - Structen som får innlest sine data
 */
void ansattLesJobbingFraFil(FILE* inn, struct Ansatt* a) { /* LAG INNMATEN */ }

/**
 * Oppgave 2F - Statistikk over de ansattes jobbing.
 */
void jobbStatistikk() { /* LAG INNMATEN OG ANNEN KODE/FUNKSJON(ER) */ }

/**
 * Oppgave 2G - Leser ALLE ansatte fra fil.
 *
 * @see ansattLesFraFil(...)
 */
void lesAnsatteFraFil() { /* LAG INNMATEN */ }

/**
 * Oppgave 2G - Leser ALT om EN ansatt fra fil.
 *
 * @param inn - Filen det skal leses fra
 * @param a - Structen som får innlest sine data
 */
void ansattLesFraFil(FILE* inn, struct Ansatt* a) { /* LAG INNMATEN */ }

```