NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1001 – Grunnleggende programmering

Faglig kontakt under eksamen: **Frode Haug**

Eksamensdato: **13.august 2025**

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

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

Annen informasjon:

Målform/språk: **Bokmål**

Antall sider (inkl. forside): **9**

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig **X** **2-sidig** ☐

sort/hvit **X** **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> // strchr(s, c) - returnerer peker til første forekomst
                    //               av 'c' i stringen 's', evt. NULL.

char txt[]    = "Cancelada-AgiaMarina-Trogir-Lokken-Bornholm-Kragero-Risor";

int main() {
  char *t = txt, *t2, *t3, *t4;

  while (*t)  {
    if (*t == 'i') printf("%c ", *(t+1));
    t++;
  }
  printf("\n");

  t = txt;   t = strchr(t, 'o');
  while (t != NULL) {
    printf("%c ", *(t-1));
    t = strchr(++t, 'o');
  }
  printf("\n");

  t = txt;
  for (int i = 0;  i < 5;  i++, t++) t = strchr(t, '-');
  while (*t != '-') printf("%c", *t++);
  printf("\n");

  t = txt+6;  t2 = txt+52;
  while (t < t2)  {
    printf("%c %c ", *t, *t2);
    t += 4;  t2 -= 8;
  }
  printf("\n");

  t = txt+22;  t2 = txt+37;  t3 = &txt[49];  t4 = &txt[strlen(txt)-1];
  printf("%c %c %c %c ", *t, *t2, *t3, *t4);
  while (*t == *t2  &&  *t3 == *t4)  {
    t++;  t2--;  t3++;  t4--;
  }
  printf("%c %c %c %c\n", *t, *t2, *t3, *t4);

  return 0;
}
```

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

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

char navn[][10] = { "Oxford", "Watford", "Luton", "Fulham", "Tottenham" };
int  dato[]     = { 18, 18, 21, 19, 19 };

struct Lag  {
  char nvn[20];   int  dato;
};

struct Lag lag[5];

void S25Funk1()  {
  for (int i = 4;  i >= 0;  i--) {
      strcpy(lag[4-i].nvn, navn[i]);   lag[4-i].dato = dato[i];
  }
}

void S25Funk2(const struct Lag s)  {
  printf("%s-%i  ", s.nvn, s.dato);
}

bool S25Funk3(const int i)  {
  return (strstr(lag[i].nvn, "ford") != NULL);
}

void S25Funk4(const struct Lag* lag1, const struct Lag* lag2)  {
  if (lag1->dato == lag2->dato  &&
      !strncmp(lag1->nvn+3, lag2->nvn+6, 3))
     printf("Jada\n");  else  printf("Neida\n");
}

void S25Funk5()  {
  struct Lag l;
  for (int i = -1;  i <= 1;  i++)  {
      l = lag[0];
      for (int j = 0;  j < 4;  j++) lag[j] = lag[j+1];
      lag[4] = l;
}  }

int S25Funk6(const char c1, const char c2)  {
  int n = 0;
  for (int i = 0;  i <= 4;  i++)  {
      if (strchr(lag[i].nvn, c1) != NULL) n++;
      if (strchr(lag[i].nvn, c2) != NULL) n++;
  }
  return n;
}

int main()  {
  S25Funk1();  S25Funk2(lag[4]);  S25Funk2(lag[2]);  printf("\n");
  printf("%i %i %i\n", S25Funk3(2), S25Funk3(4), S25Funk3(3));
  S25Funk4(&lag[1], &lag[0]);
  S25Funk5();   S25Funk2(lag[0]);  S25Funk2(lag[2]);  printf("\n");
  printf("%i %i\n", S25Funk6('a', 'o'),  S25Funk6('r', 'm'));
  return 0;
}
```

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2h) *nøye*, før du begynner å besvare noe som helst. Studér vedlegget, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `#define`, structen med datamedlemmer, (ferdiglagde) funksjoner (spesielt `hentDag()` og `hentTime()`), global variabel og `main()` og kommentarer i koden. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`.
<u>Bruk *alt* dette *svært* aktivt.</u>

Det skal holdes orden på antall besøk/visitasjoner på en legevakt, dvs. hvilke ukedager og timer/klokkeslett dette skjer.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gDogn`. I denne er *alltid alle* indeksene fra 0 til `DOGN`-1 i bruk. Vedlegget angir også hvilke datamedlemmer structene inneholder.
I `antall` er *alltid alle* indeksene 0 til `TIMER`-1 i bruk.
*Vedlegget inneholder alt du trenger av structer, datamedlemmer og globale variable for å løse denne eksamensoppgaven. Dessuten er **prototyper** for **alle** funksjoner også ferdig deklarert/definert.*

Oppgaven

a) **8** **Skriv innmaten til** `void skrivAlleDogn()` **og**
`void dognSkrivData(const struct Dogn* dogn)`
Den første funksjonen sørger for at det blir gått igjennom alle døgnene. For hvert døgn (den andre funksjonen) skrives kun døgnets navn ('Mandag', 'Tirsdag', …..'Lørdag', 'Søndag') og *totalt antall* legevaktbesøk i det aktuelle døgnet, dvs. *summen av alle verdiene* i døgnets `antall`.

b) **8** **Skriv innmaten til** `void skrivEttDogn()` **og**
`void dognSkrivAlt(const struct Dogn* dogn)`
Den første funksjonen spør brukeren om et døgnnummer (1=Mandag, 2=Tirsdag, … 7=Søndag). Deretter skrives *alt* om dette døgnet ut på skjermen vha. den andre funksjonen.
Dvs. *alle* structens datamedlemmer skrivs ut. En verdi pr.linje (altså totalt 26 verdier/linjer).

c) **6** **Skriv innmaten til** `void nyttBesok()` **og**
`void dognRegistrerBesok(struct Dogn* dogn)`
Den første funksjonen gjør det samme som den første i oppgave 2b. Den andre funksjonen spør brukeren om et klokkeslett (0-23) og øker antall besøk for den aktuelle timen med 1 (en).

d) **8** **Skriv innmaten til** `void nyttBesok2()` **og**
`void dognRegistrerBesok2(struct Dogn* dogn)`
Disse to funksjonene gjør til sammen det samme som de to i oppgave 2c, bare at brukeren *ikke* spørres om noe. I stedet brukes ferdiglagde funksjoner for å hente dagen og timen.
Begge disse verdiene blir også skrevet ut på skjermen.

e) **10** **Skriv innmaten til** `void naarFlestBesok()`

Funksjonen skal skrive ut navnet på ukedagen, timen og antallet det er flest besøk på legevakten. Er det flere like som er flest, skrives den første ut.

NB: Du trenger *ikke* å lage egne funksjon(er) som aksesserer struct-medlemmer, men bare gjøre det direkte fra koden i denne funksjonen.

f) **12** **Skriv innmaten til** `void naarFlestBesok2()`

Funksjonen teller opp antall besøk som *totalt* er pr.klokketime, *uavhengig av ukedag*. Den skriver ut det høyeste totalantallet, og *alle* de timene (en eller flere) som har dette antallet.

NB: Du trenger *ikke* å lage egne funksjon(er) som aksesserer struct-medlemmer, men bare gjøre det direkte fra koden i denne funksjonen.

g) **8** **Skriv innmaten til** `void skrivDognTilFil()` **og**
`void dognSkrivTilFil(FILE* ut, struct Dogn* dogn)`

Funksjonene sørger til sammen for at *alle* døgnene blir skrevet ut til filen «LEGEVAKT.DTA». **Filformatet** bestemmer du helt selv, men ***skal*** **oppgis som en del av besvarelsen**.

h) **10** **Skriv innmaten til** `void lesDognFraFil()` **og**
`void dognLesFraFil(FILE* inn, struct Dogn* dogn)`

Funksjonene sørger til sammen for at *alle* døgnene blir lest inn fra filen «LEGEVAKT.DTA», etter det formatet du selv bestemte i oppgave 2g.

Annet (klargjørende):

- Ordene «døgn» og «dag» brukes synonymt i all teksten ovenfor og i koden (vedlegget). Det samme gjør ordene «besøk», «legevaktbesøk» og «visitasjon», og «time» og «klokkeslett».
- 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 aktuell deloppgave.
- **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 includer brukt i hele høst er tilgjengelig.

Håper det blir lenge til neste legevaktbesøk!
FrodeH

Vedlegg til PROG1001, august 2025: Halvferdig programkode

```
#include <stdio.h>              //  printf, FILE
#include <stdlib.h>             //  sizeof, malloc
#include <string.h>             //  strcpy, strlen
#include <time.h>               //  Noen tidsfunksjoner for å få tak i dag/time
#include "LesData.h"            //  Verktøykasse for lesing av diverse data

#define   DOGN            7     ///<  Antall dager i en uke.
#define   TIMER          24     ///<  Antall timer i et døgn.
#define   STRLEN        120     ///<  Max. tekstlengde.

/**
 *  Døgn (med døgnets/dagens navn, beskrivelse av det,
 *        og antall besøk pr time i det aktuelle døgnet).
 */
struct Dogn  {
  char* navn;                   //  Dagen/døgnets navn.
  char* beskrivelse;            //  Beskrivelse av døgnet.
  int   antall[TIMER];          //  Antall besøk for hver klokketime i døgnet.
};

void skrivMeny();                                                   //  Ferdig-
int  hentDag();                                                     //    laget:
int  hentTime();                                                    //
void skrivAlleDogn();                                               //  Oppgave 2A
void dognSkrivData(const struct Dogn* dogn);                        //  Oppgave 2A
void skrivEttDogn();                                                //  Oppgave 2B
void dognSkrivAlt(const struct Dogn* dogn);                         //  Oppgave 2B
void nyttBesok();                                                   //  Oppgave 2C
void dognRegistrerBesok(struct Dogn* dogn);                         //  Oppgave 2C
void nyttBesok2();                                                  //  Oppgave 2D
void dognRegistrerBesok2(struct Dogn* dogn);                        //  Oppgave 2D
void naarFlestBesok();                                              //  Oppgave 2E
void naarFlestBesok2();                                             //  Oppgave 2F
void skrivDognTilFil();                                             //  Oppgave 2G
void dognSkrivTilFil(FILE* ut, struct Dogn* dogn);                  //  Oppgave 2G
void lesDognFraFil();                                               //  Oppgave 2H
void dognLesFraFil(FILE* inn, struct Dogn* dogn);                   //  Oppgave 2H


struct Dogn* gDogn[DOGN];       ///<  Alle døgnene i registeret.

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

  lesDognFraFil();                                                  //  Oppgave 2H

  skrivMeny();
  kommando = lesChar("\nValg");

  while (kommando != 'Q') {
    switch (kommando) {
      case 'A':  skrivAlleDogn();          break;                   //  Oppgave 2A
      case 'E':  skrivEttDogn();           break;                   //  Oppgave 2B
      case 'N':  nyttBesok();              break;                   //  Oppgave 2C
      case 'B':  nyttBesok2();             break;                   //  Oppgave 2D
      case 'F':  naarFlestBesok();         break;                   //  Oppgave 2E
      case 'T':  naarFlestBesok2();        break;                   //  Oppgave 2F
      default:   skrivMeny();              break;
    }
    kommando = lesChar("\nValg");
  }
```

```
  skrivDognTilFil();                                          //  Oppgave 2G

  return 0;
}



/**
 *  Avleser maskinens systemklokke, og returnerer ukedagen (man=1, ... søn=7).
 *
 *  @return   Ukedagen (man=1, tirs=2, ... lør=6, søn=7)
 */
int hentDag()  {
  time_t result = time(0);                    //  Henter maskinens tidspunkt.
  struct tm tid = *localtime(&result);        //  Konverterer inn i struct.
  return ((tid.tm_wday == 0) ? 7 : tid.tm_wday);   //  Returnerer 1-7.
}



/**
 *  Avleser maskinens systemklokke, og returnerer hvilken time det er (0-23).
 *
 *  @return   Hvilken time det er (0-23).
 */
int hentTime()  {
  time_t result = time(0);                    //  Henter maskinens tidspunkt.
  struct tm tid = *localtime(&result);        //  Konverterer inn i struct.
  return (tid.tm_hour);                       //  Returnerer 0-23.
}



/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
  printf("\nFOLGENDE KOMMANDOER ER LOVLIG:\n");
  printf("\tA   = skriv Alle dogn\n");
  printf("\tE   = skriv Ett dogn\n");
  printf("\tN   = Nytt besok (versjon 1)\n");
  printf("\tB   = nytt Besok (versjon 2)\n");
  printf("\tF   = hvilket dogn og time med Flest besok\n");
  printf("\tT   = hvilken Time med totalt flest besok (uavhengig av dogn)\n");
  printf("\tQ   = Quit/avslutt\n");
}



/**
 *  Oppgave 2A - Skriver ALLE døgnene og TOTALT antall besøk pr døgn.
 *
 *  @see   dognSkrivData(...)
 */
void skrivAlleDogn()  {                          /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2A - Skriver døgnnavn og TOTALT antall besøk det døgnet.
 *
 *  @param  dogn  -  Døgnet det skrives ut noen data for
 */
void dognSkrivData(const struct Dogn* dogn)  {       /*    LAG INNMATEN      */  }
```

```
/**
 *  Oppgave 2B - Skriver ALT om ETT gitt døgn.
 *
 *  @see   dognSkrivAlt(...)
 */
void skrivEttDogn()  {                                  /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2B - Skriver ALT om ett døgn.
 *
 *  @param  dogn  -  Døgnet det skrives ut ALT om
 */
void dognSkrivAlt(const struct Dogn* dogn)  {           /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2C - Leser inn et nytt besøk på legevakten.
 *
 *  @see   dognRegistrerBesok(...)
 */
void nyttBesok()  {                                     /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2C - Registrerer ETT besøk i aktuelt døgn.
 *
 *  @param  dogn  -  Døgnet det registreres ETT besøk i
 */
void dognRegistrerBesok(struct Dogn* dogn)  {           /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2D - Nytt besøk registreres AUTOMATISK ved å avlese systemklokka.
 *
 *  @see   dognRegistrerBesok2(...)
 */
void nyttBesok2()  {                                    /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2D - Registrerer AUTOMATISK ETT besøk i aktuelt døgn.
 *
 *  @param  dogn  -  Døgnet det registreres ETT besøk i
 */
void dognRegistrerBesok2(struct Dogn* dogn)  {          /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2E - Finner og skriver dagen og timen med flest legevaktbesøk.
 */
void naarFlestBesok()  {                                /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2F - Finner og skriver timen TOTALT i HELE uka
 *               med flest legevaktbesøk.
 */
void naarFlestBesok2()  {                               /*    LAG INNMATEN      */  }
```

```
/**
 *  Oppgave 2G - Skriver ALLE døgnene ut til fil.
 *
 *  @see   dognSkrivTilFil(...)
 */
void skrivDognTilFil()  {                              /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2G - Skriver ALT om ETT døgn ut til fil.
 *
 *  @param  inn   -  Filen det skal skrives ut til
 *  @param  dogn  -  Døgnet som får skrevet ut sine data
 */
void dognSkrivTilFil(FILE* ut, struct Dogn* dogn)  { /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2H - Leser ALLE døgnene inn fra fil.
 *
 *  @see   dognLesFraFil(...)
 */
void lesDognFraFil()  {                                /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2H - Leser ALT om ETT døgn inn fra fil.
 *
 *  @param  inn   -  Filen det skal leses inn fra
 *  @param  dogn  -  Døgnet som får innlest sine data
 */
void dognLesFraFil(FILE* inn, struct Dogn* dogn)  {  /*    LAG INNMATEN      */  }
```