NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1001 – Grunnleggende programmering

Faglig kontakt under eksamen:	**Frode Haug**
Eksamensdato:	**8.desember 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):	**8**

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[]  = "Svolvaer-Stamsund-Stokmarknes-Sortland-Sakrisoy-Sorvaagen";
char txt2[] = "aet";


int main() {
  char *t;
  int  i,  n = 0;

  for (i = 0;  i < 56;  i+=8)  printf("%c ", txt[i]);
  printf("\n");

  t = txt;
  while (*t != '\0')   {
    if (*t == 'S') printf("%c ", *(t+2));    ++t;
  }
  printf("\n");

  for (i = 0;  i <=52;  i++)
      if (!strncmp(txt+i, "St", 2)  ||  !strncmp(txt+i, "Sor", 3))
         printf("%c ", txt[i+3]);
  printf("\n");

  for (i = 0;  i < 3;  i++)  {
      t = strchr(txt, txt2[i]);
      printf("%c ", *(t+2));
  }
  printf("\n");

  for (i = 0;  i < 3;  i++)  {
      t = strchr(txt, txt2[i]);
      while (t != NULL)  {
        n++;
        t = strchr(++t, txt2[i]);
      }
  }
  printf("%i\n", n);

  return 0;
}
```

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

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

struct Lag  {
   char navn[30];
   char stadion[30];
};

char txt1[][30] = { "Arsenal",  "Portsmouth",   "Ipswich",      "Brentford" };
char txt2[][30] = { "Emirates", "Fratton Park", "Portman Road", "Community" };
struct Lag lagene[4];

void H25Funk1(const struct Lag l, const bool s)  {
  printf("%s: %s", l.navn, l.stadion);    if (s) printf(" Stadium");
}

void H25Funk2(const char ch)  {
  for (int i = 0;  i < 4;  i++)
      if (strchr(txt1[i], ch) != NULL  &&  strchr(txt2[i], ch) != NULL)
         printf("%i ", i);
}

bool H25Funk3(const int n)  {
  return (strlen(lagene[n].navn) < strlen(lagene[n].stadion));
}

char* H25Funk4(const int i, const int j)  {
  char* t = (char*) malloc(40 * sizeof(char));
  strcpy(t, txt1[i]);   strcat(t, " - ");  strcat(t, txt2[j]);
  return t;
}

int main()  {

  for (int i = 0;  i < 4;  i++)
      if (i % 2 == 0) printf("%s  ", txt1[i]);
      else            printf("%s  ", txt2[i]);
  printf("\n");


  for (int i = 3;  i >= 0;  i--)  {
      strcpy(lagene[i].navn,    txt1[3-i]);
      strcpy(lagene[i].stadion, txt2[3-i]);
  }
  H25Funk1(lagene[1], false);  printf(" ... ");
  H25Funk1(lagene[0], true);   printf("\n");

  H25Funk2('o');  printf("\n");

  printf("%i %i\n", H25Funk3(3), H25Funk3(0));

  printf("%s\n", H25Funk4(2, 1));

  return 0;
}
```

Oppgave 2 (70%)

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

En litt større sosial vennegjeng ønsker å samles/treffe hverandre ofte. For å få til dette blir det *alltid* den første dagen i måneden lagt inn *alle* dagene de *ulike* personene har mulighet i den kommende måneden. Dette gjør at programmet vi skal lage kan komme med forslag til ulike dager de kan treffes.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gDeltagere` og `gAntallDeltagere`.
I arrayen er indeksene fra 0 til `gAntallDeltagere-1` i bruk. Vedlegget angir også hvilke datamedlemmer structene inneholder. I `dag` er *alltid alle* indeksene 0 til `ANTDAG-1` i bruk.
I denne eksamensoppgaven lar vi det for enkelthets skyld *alltid være 31 dager i hver måned*.
*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 skrivAlleDeltagere()` og `void deltagerSkrivData(const struct Deltager* deltager)`

Den første funksjonen sørger for at det blir gått igjennom alle aktuelle deltagere, og skriver ut deres nummer (fra 1-en og oppover, selv om den første har indeks nr.0-null). Den andre funksjonen skriver ut *alle* (unntatt `dag`-arrayens innhold) structens datamedlemmer på *en linje*.

b) 10 Skriv innmaten til `void nyDeltager()` og `void deltagerLesData(struct Deltager* deltager)`

Den første funksjonen sjekker først om det er plass til flere deltagere. Om så *ikke* er tilfelle, kommer det en melding. I motsatt fall opprettes og legges det inn en ny deltager, og dennes nummer skrives ut på skjermen. *Alle* deltagerens data (unntatt innholdet i `dag`) leses inn vha. den andre funksjonen. Den sørger i stedet for (ved å bruke den andre funksjonen i oppgave 2d) at *hele* arrayen `dag` settes til `false`.

c) 10 Skriv innmaten til `void fjernDeltager()` og `void deltagerFrigi(struct Deltager* deltager)`

Den første funksjonen spør først om et lovlig deltagernummer (inkludert nr.0). Om brukeren velger 0 (null), kommer det en melding om det, og ingenting skjer. I motsatt fall *frigis all* brukt memory ifm. structen (bl.a. ved å kalle den andre funksjonen), og den siste deltageren flyttes til plassen/indeksen der den slettede har vært, og antall deltagere telles ned med 1-en.

d) **6** **Skriv innmaten til** `void nullstillAlle()` **og**
`void deltagerNullstill(struct Deltager* deltager)`
Den første funksjonen sørger for at *alle* deltagerne får satt sin array `dag` til *kun* å inneholde `false` (vha. den andre funksjonen).

e) **12** **Skriv innmaten til** `void lesDager()` **og**
`void deltagerLesDager(struct Deltager* deltager)`
Den første funksjonen ber om et lovlig deltagernummer, og kaller deretter den andre funksjonen. Denne funksjonen sørger først for at *hele* deltagerens `dag` settes til `false`. Deretter spør den om dagnumre helt til brukeren skriver 0 (null). Så lenge dagnummeret er ulikt 0 (null) markeres den aktuelle dagen med `true`. Til slutt gås det gjennom `dag`, og alle dagnumrene (indeksene) vedkommende kan møte blir skrevet ut på skjermen.

f) **12** **Skriv innmaten til** `void finnMoteDager()`
og evt. annen kode/funksjon(er) som trengs for å skrive ut på skjermen:
- *alle* dager *alle* kan møte
- hvilke dager *minst halvparten*, men *ikke alle* kan møte.
  I tillegg til dagnummeret skrives antallet som kan møte.

g) **12** **Skriv innmaten til** `void lesFraFil()` **og**
`void deltagerLesFraFil(FILE* inn, struct Deltager* deltager)`
Funksjonene sørger til sammen for at *alle* rutene blir lest inn fra filen «SAMLING.DTA».
Filformatet bestemmer du helt selv, men ***skal*** **oppgis som en del av besvarelsen**.

Klargjøring og forutsetninger

- I praksis brukes programmet på følgende måte: ***Den første dagen i hver måned***
  (på morgenen/formiddagen) utføres **2d** (en gang) etterfulgt av **2e** (en gang for hver av deltagerne). Deretter gjøres **2f**. Deltagerne kan så varsles (på en eller annen elektronisk måte) om når det blir ett eller flere møter/treff den forestående måneden. Dette avgjør den som kjører programmet, ettersom hun/han finner dag(er) der alle eller mange kan møtes/treffes.
- Å fjerne (**2c**) eller legge inn en ny deltager (**2b**) kan utføres når som helst i løpet av måneden.
- Husk at i arrayene `gDeltagere` og `dag` brukes altså indeksene 0 (null) og til henholdsvis `gAntallDeltagere/ANTDAG - 1`. Men all dialog med brukeren (innlesning og utskrift) foregår på formen 1 (en) til `gAntallDeltagere/ANTDAG`.
- Å skrive til fil er altså ikke en del av denne eksamensoppgaven. Men dette må også selvsagt gjøres.
- 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.

Lykke til – og antagelig treffes vi en dag!

FrodeH

Vedlegg til PROG1001, desember 2025: Halvferdig programkode

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

#define   MAXDELTAGERE  20      ///<  Max. antall deltagere i gruppen.
#define   ANTDAG        31      ///<  Max. antall dager i en måned.
const int STRLEN   =    80;     ///<  Max. tekstlengde.

/**
 *  Deltager (med navnet, mailadresse, telefon og hvilke dager kan delta).
 */
struct Deltager {
  char* navn;                   //  Deltagerens navn.
  char* mail;                   //  Mailadresse.
  int   tlf;                    //  Telefon (fast eller mobil).
  bool  dag[ANTDAG];            //  Kan (ikke) på dag nr.'i'
};                              //       der indeksene 0 til ANTDAG-1 brukes.

void skrivMeny();                                                   //  Ferdiglaget
void skrivAlleDeltagere();                                          //  Oppgave 2A
void deltagerSkrivData(const struct Deltager* deltager);            //  Oppgave 2A
void nyDeltager();                                                  //  Oppgave 2B
void deltagerLesData(struct Deltager* deltager);                    //  Oppgave 2B
void fjernDeltager();                                               //  Oppgave 2C
void deltagerFrigi(struct Deltager* deltager);                      //  Oppgave 2C
void nullstillAlle();                                               //  Oppgave 2D
void deltagerNullstill(struct Deltager* deltager);                  //  Oppgave 2D
void lesDager();                                                    //  Oppgave 2E
void deltagerLesDager(struct Deltager* deltager);                   //  Oppgave 2E
void finnMoteDager();                                               //  Oppgave 2F
void lesFraFil();                                                   //  Oppgave 2G
void deltagerLesFraFil(FILE* inn, struct Deltager* deltager);       //  Oppgave 2G

struct Deltager* gDeltagere[MAXDELTAGERE];    ///<  Array med Deltagere.
int gAntallDeltagere = 0;                     ///<  Antall deltagere hittil.

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

   lesFraFil();                                             //  Oppgave 2G

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

   while (kommando != 'Q')  {
     switch (kommando)  {
       case 'S':  skrivAlleDeltagere();    break;       //  Oppgave 2A
       case 'N':  nyDeltager();            break;       //  Oppgave 2B
       case 'F':  fjernDeltager();         break;       //  Oppgave 2C
       case 'U':  nullstillAlle();         break;       //  Oppgave 2D
       case 'L':  lesDager();              break;       //  Oppgave 2E
       case 'D':  finnMoteDager();         break;       //  Oppgave 2F
       default:   skrivMeny();             break;
     }
     kommando = lesChar("\nValg");
   }

  return 0;
}
```

```
/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny()  {
  printf("\n\nFOLGENDE KOMMANDOER ER LOVLIG:\n");
  printf("\tS   = Skriv alle deltageres hoveddata\n");
  printf("\tN   = Ny deltager\n");
  printf("\tF   = Fjern deltager\n");
  printf("\tU   = nUllstill dagene for ALLE deltagerne\n");
  printf("\tL   = Les dager for EN deltager\n");
  printf("\tD   = finn felles Dager for aktuelle moter/treff\n");
  printf("\tQ   = Quit/avslutt\n");
}


/**
 *  Oppgave 2A - Skriver ALLE deltagernes hoveddata.
 *
 *  @see   deltagerSkrivData(...)
 */
void skrivAlleDeltagere()  {                                /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2A - Skriver ALLE data (unntatt 'dag') for EN deltager ut på skjerm.
 *
 *  @param   deltager  -  Deltageren det skrives ut alle hoveddata for
 */
void deltagerSkrivData(const struct Deltager* deltager)  { /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2B - Legger (om mulig) inn EN ny deltager.
 *
 *  @see   deltagerLesData(...)
 */
void nyDeltager()  {                                        /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2B - Leser inn alle data om EN deltager (unntatt 'dag').
 *
 *  @param   deltager  -  Deltageren som får sine data innlest
 *  @see     deltagerNullstill(...)
 */
void deltagerLesData(struct Deltager* deltager)  {          /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2C - Fjerner EN deltager.
 *
 *  @see   deltagerFrigi(...)
 */
void fjernDeltager()  {                                     /*   LAG INNMATEN   */ }


/**
 *  Oppgave 2C - Frigir all allokert memory inni EN deltager-struct.
 *
 *  @param   deltager  -  Deltageren som får sin memory frigitt
 */
void deltagerFrigi(struct Deltager* deltager)  {            /*   LAG INNMATEN   */ }
```

```
/**
 *  Oppgave 2D - Nullstiller (til 'false') ALLE deltagernes 'dag'-array.
 *
 *  @see   deltagerNullstill(...)
 */
void nullstillAlle()  {                                    /*   LAG INNMATEN   */ }



/**
 *  Oppgave 2D - Nullstiller (til 'false') HELE en deltagers 'dag'-array.
 *
 *  @param   deltager  -  Deltageren som får sin array 'dag' satt til 'false'
 */
void deltagerNullstill(struct Deltager* deltager)  {       /*   LAG INNMATEN   */ }



/**
 *  Oppgave 2E - Leser EN deltagers mulige møtedager.
 *
 *  @see   deltagerLesDager(...)
 */
void lesDager()  {                                         /*   LAG INNMATEN   */ }



/**
 *  Oppgave 2E - Leser aktuelle møtedager for EN deltager.
 *
 *  @param    deltager  -  Deltageren som får akyuelle møtedager lest inn
 *  @see      deltagerNullstill(...)
 */
void deltagerLesDager(struct Deltager* deltager)  {        /*   LAG INNMATEN   */ }



/**
 *  Oppgave 2F - Finner dager der flest kan møtes/samles.
 */
void finnMoteDager()  {                                    /*   LAG INNMATEN   */ }



/**
 *  Oppgave 2G - Leser ALLE deltagerne fra fil.
 *
 *  @see   deltagerLesFraFil(...)
 */
void lesFraFil()  {                                        /*   LAG INNMATEN   */ }



/**
 *  Oppgave 2G - Leser ALT om EN deltager inn fra fil.
 *
 *  @param  inn       -  Filen det skal leses inn fra
 *  @param  deltager  -  Deltageren som får innlest sine data
 */
void deltagerLesFraFil(FILE* inn, struct Deltager* deltager)  {
                                                           /*   LAG INNMATEN   */ }
```