NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1001
– Grunnleggende programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**21.desember 2023**
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>


char txt[] =
       "LEICESTER-LEEDS-BIRMINGHAM-IPSWICH-NOTTINGHAM-ASTONVILLA";


int main() {
    int i = 35,  j = i/6,  k = i%8;

    while (txt[i] != '\0')  {
        if (txt[i] == 'T') printf("%c ", txt[i-2]);
        ++i;
    }
    printf("\n");

    for (i = 10;  txt[i] != 'P';  i+= k)
        printf("%c ", txt[i]);
    printf("\n");

    i = 19;
    do {
       printf("%c ", txt[++i]);
    }  while (txt[i-k] != txt[i+k]);
    printf("\n");

    char* t = &txt[0], *s = t+35;
    while (*t++ != *s--)
        printf("%c ", txt[j++]);
    printf("\n");

    t = strstr(txt, "HAM");
    while (t)  {
        printf("%c%c%c%c ", *(t-4),*(t-3),*(t-2),*(t-1));
        t++;  t = strstr(t, "HAM");
    }
    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>

char tekst[][6] = { "Jada", "Neida" };

struct Person  {
  char fNavn[40], eNavn[40];
  int  alder;
};

void H23Funk1(const struct Person person)  {
  printf("%s, %s (%i)", person.eNavn, person.fNavn, person.alder);
}

bool H23Funk2(const struct Person person)  {
  return (person.alder < 20  ||  person.alder >= 51);
}

bool H23Funk3(const struct Person* person)  {
  return (strstr(person->eNavn, "e")  &&  strstr(person->fNavn, "a"));
}

struct Person H23Funk4(const struct Person* p1, const struct Person* p2)  {
  struct Person p;
  strcpy (p.fNavn, p1->fNavn);  strcpy (p.eNavn, p2->eNavn);
  p.alder = p1->alder + p2->alder;
  strcat(p.fNavn, "-");  strcat(p.fNavn, tekst[H23Funk2(p)]);
  return p;
}

char H23Funk5(const struct Person p1, const struct Person* p2)  {
  if (!strncmp(&(p1.fNavn[2]), &(p2->fNavn[1]), 3))  {
     if (p1.alder-31 >=  p2->alder)  {
        return ((strlen(p2->eNavn) < strlen(p1.eNavn)) ? 'S' : 'J');
     } else
        return 'T';
  } else  return 'F';
}

int main()  {
  struct Person person1 = { "Anne",  "Andersen", 19 },
                person2 = { "Frank", "Frantzen", 53 },
                person3 = { "Ida",   "Iversen",  20 },
                person4 = { "Marit", "Mentzoni", 22 },
                person5 = { "Sanne", "Syversen", 51 };

  H23Funk1(person2);   printf(" og ");   H23Funk1(person4);   printf("\n");

  printf("%i %i %i\n", H23Funk2(person1), H23Funk2(person3), H23Funk2(person5));

  printf("%i %i\n", H23Funk3(&person1), H23Funk3(&person2));

  H23Funk1(H23Funk4(&person3, &person4));   printf("\n");

  printf("%c\n", H23Funk5(person5, &person1));

  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 datamedlemmer, funksjoner, globale variable, `main()` og *tre ferdiglagde funksjoner*. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`. Bruk *alt* dette *svært* aktivt.

Det holdes orden på ulike observasjoner av ulike fugler (arter/typer) i *ett og samme* kalenderår (og som *ikke* er skuddår).

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gFugler`. I denne er indeksene fra 0 til `gAntallFugler-1` i bruk. Vedlegget angir også hvilke datamedlemmer structen inneholder.
NB: Legg merke til hvilket format observasjonene lagres på (MMDD), men skrives ut på skjermen som: DD/MM. *Vedlegget inneholder alt du trenger av structer, datamedlemmer og globale variable for å løse denne eksamensoppgaven. Dessuten er **prototyper** for de fleste funksjoner også ferdig deklarert/definert.*

Oppgaven

a) **12** **Skriv innmaten til** `void nyFugl()`
`void fuglSettData(struct Fugl* fugl, char* nvn)` **og**
`void skrivDato(const int dato)`

Den første funksjonen kommer med en egen melding om det allerede er fullt med fugler. I motsatt fall leses et aktuelt fuglenavn. Finnes denne allerede, kommer det også en egen melding, og evt. allokert memory frigis igjen. Ellers opprettes en ny `Fugl`. I den andre funksjonen lagres dens navn, og antall observasjoner settes til 0 (null). Den tredje funksjonen (som har ingenting med de ovenfor å gjøre, men bør nok brukes i funksjoner nedenfor) får inn en dato på formatet MMDD, og skriver kun dette ut på skjermen som DD/MM (dag og måned med '/' imellom).

b) **8** **Skriv innmaten til** `void skrivAlleFugler()` **og**
`void fuglSkrivNoe(const struct Fugl* fugl)`

Den første funksjonen går igjennom alle registrerte fugler og skriver deres nummer (nummerert fra 1 (en) og oppover) og *alle* fuglenes data vha. den andre funksjonen. Denne funksjonen skriver *kun* ut (på *en* linje) fuglens navn og antall observasjoner (men altså *ikke* datoene for alle observasjonene – det gjøres i neste oppgave).

c) **10** **Skriv innmaten til** `void skrivEnFugl()` **og**
`void fuglSkrivAlt(const struct Fugl* fugl)`

Den første funksjonen kommer med en egen melding om det ennå ikke er registrert noen fugler. I motsatt fall leses ut fuglenavn. Finnes denne *ikke*, kommer det også en egen melding. Ellers skrives alt om fuglen ut på skjermen. Dvs. antall observasjoner og *alle* datoene (på formen DD/MM) for alle observasjonene (navnet er overflødig, for brukeren jo skrevet det). Datoene *skal* være adskilt vha. «, » (komma og to blanke), *unntatt etter den siste datoen*.

d) **10** ***Angi*** **innmaten til** `void nyObservasjon()` **og**
skriv innmaten til `void fuglNyObservasjon(struct Fugl* fugl)`
Den første funksjonen gjør akkurat det samme som den første i oppgave 2C, bare at den andre funksjonen rett ovenfor kalles i stedet. **Forklar *kun* hva endringen i koden vil bli.**
Den andre funksjonen kommer med en melding om det er fullt med observasjoner allerede.
I motsatt fall leser den et dagnummer (1-31) og en måned (1-12). Dette gjør den inntil datoen er en lovlig/gyldig dato (bruk ferdiglaget funksjon). Datoen bør så gjøres om til formatet MMDD. Datoen legges så inn som en ny observasjon av den aktuelle fuglen, men *kun* om datoen er *større enn den sist registrerte*. Ellers kommer det en melding som bl.a. inneholder hva denne siste registrerte datoen er (på formen DD/MM). Husk å spesialbehandle den aller første datoen som blir registrert.
Dvs. vi skal altså *ikke* registrere flere observasjoner på samme dato for samme fugl.
Dessuten vil datoene på denne måten bli i stigende datorekkefølge i året.

e) **12** **Skriv innmaten til** `void dagenMedFlestObservasjoner()`
og all annen kode/funksjon(er) som trengs for å finne og skrive ut hvilken dato (på formen DD/MM) det *totalt* er gjort flest observasjoner av fugler.
Har flere datoer dette samme antallet, skrives bare den første datoen ut på skjermen.
NB: Husk å følge tankegangen i måten koden ellers er skrevet på, slik at *ikke en* funksjon *alene* gjør *all* jobben, også å aksessere *alle* structenes datamedlemmer.

f) **8** **Skriv innmaten til** `void skrivTilFil()` **og**
`void fuglSkrivTilFil(FILE* ut, const struct Fugl* fugl)`
Funksjonene sørger til sammen for at alle fuglene blir skrevet til filen «FUGLER.DTA».
`gAntallFugler` skal *ikke* ligge på filen. **Filformatet** ellers bestemmer du helt selv, men ***skal*** **oppgis som en del av besvarelsen**.

g) **10** **Skriv innmaten til** `void lesFraFil()` **og**
`void fuglLesFraFil(FILE* inn, struct Fugl* fugl, const char* nvn)`
Funksjonene sørger til sammen for at *alle* fuglene blir lest inn fra filen «FUGLER.DTA», etter det formatet du selv bestemte i forrige deloppgave.

Annet (klargjørende):

- I teksten/koden brukes stort sett betegnelsen «fugl» som kortform for «fugleart» / «fugletype».
- 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 med observasjonen av eksamensoppgaven!
FrodeH

Vedlegg til PROG1001, desember 2023: Halvferdig programkode

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

#define   MAXFUGLER      200    ///<  Max. antall fuglearter/-typer.
#define   ANTDAGER       365    ///<  Dagantall i ETT år (ignorerer skuddår).
const int STRLEN  =       80;   ///<  Max. tekstlengde.

/**
 *  Fugl (med navn, antall observasjoner og datoene for disse observasjonene).
 */
struct Fugl  {                  //  En fuglearts/-types:
  char* navn;                   //    - navn
  int   antObs;                 //    - antall observasjoner i ETT kalenderår
  int   observasjoner[ANTDAGER]; //    - datoer for observasjonene.
};                              //        Lagres på formen: MMDD

int  dagnummer(const int dag, const int mnd);                  //  | Ferdig-
int  finnFugl(const char* nvn);                                 //  |  laget:
void skrivMeny();                                               //  |
void nyFugl();                                                  //  Oppgave 2A
void fuglSettData(struct Fugl* fugl, char* nvn);                //  Oppgave 2A
void skrivDato(const int dato);                                 //  Oppgave 2A
void skrivAlleFugler();                                         //  Oppgave 2B
void fuglSkrivNoe(const struct Fugl* fugl);                     //  Oppgave 2B
void skrivEnFugl();                                             //  Oppgave 2C
void fuglSkrivAlt(const struct Fugl* fugl);                     //  Oppgave 2C
void nyObservasjon();                                           //  Oppgave 2D
void fuglNyObservasjon(struct Fugl* fugl);                      //  Oppgave 2D
void dagenMedFlestObservasjoner();                              //  Oppgave 2E
void skrivTilFil();                                             //  Oppgave 2F
void fuglSkrivTilFil(FILE* ut, const struct Fugl* fugl);        //  Oppgave 2F
void lesFraFil();                                               //  Oppgave 2G
void fuglLesFraFil(FILE* inn, struct Fugl* fugl, const char* nvn); // Oppg.2G

int    gAntallFugler = 0;               ///<  Antall fugler hittil registrert.
struct Fugl* gFugler[MAXFUGLER];        ///<  Alle fuglene i registeret.

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

  lesFraFil();                                                  //  Oppgave 2G

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

  while (kommando != 'Q') {
    switch (kommando) {
      case 'F':  nyFugl();                    break;    //  Oppgave 2A
      case 'A':  skrivAlleFugler();           break;    //  Oppgave 2B
      case 'E':  skrivEnFugl();               break;    //  Oppgave 2C
      case 'O':  nyObservasjon();             break;    //  Oppgave 2D
      case 'D':  dagenMedFlestObservasjoner(); break;   //  Oppgave 2E
      default:   skrivMeny();                 break;
    }
    kommando = lesChar("\nValg");
  }

  skrivTilFil();                                                //  Oppgave 2F

  return 0;
}
```

```
/**
 *  Beregner (om mulig) dagnummeret i året for en gitt dato (IGNORERER skuddår).
 *
 *  @param   dag  -  Dag
 *  @param   mnd  -  Måned
 *  @return  Dagnummeret (1-365) i året (om gyldig dato) ellers 0 (null)
 */
int dagNummer(const int dag, const int mnd) {
  int dagnr;                          //  Setter opp antall dager i månedene:
  int dagerPrMnd[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

  if (mnd < 1 || mnd > 12)                return 0;  //  Ulovlig måned.
  if (dag < 1 || dag > dagerPrMnd[mnd-1]) return 0;  //  Ulovlig dag.
  dagnr = dag;
  for (int i = 0;  i < mnd-1;  i++)         //  Regner ut datoens dagnummer.
      dagnr += dagerPrMnd[i];
  return dagnr;                             //  Returnerer dagnummeret.
}


/**
 *  Prøver å finne indeksen i 'gFugler' for en navngitt fugl.
 *
 *  @param   nvn  - Søkt fugls navn
 *  @return  Indeksen for fuglen, evt. -1 om ikke ble funnet
 */
int finnFugl(const char* nvn)  {
  for (int i = 0;  i < gAntallFugler;  i++)
      if (!strcmp(gFugler[i]->navn, nvn))  return i;   //  Funn/match!

  return -1;                            //  Ikke funnet noen med dette 'nvn'.
}


/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
  printf("\nFOLGENDE KOMMANDOER ER LOVLIG:\n");
  printf("\tF   = ny Fugl\n");
  printf("\tA   = skriv Alle fuglene og antall observasjoner\n");
  printf("\tE   = skriv alt om En fugl\n");
  printf("\tO   = ny Observasjon av en fugl\n");
  printf("\tD   = Dagen med flest fugleobservasjoner\n");
  printf("\tQ   = Quit/avslutt\n");
}

/**
 *  Oppgave 2A - Legger (om mulig) inn en ny fugl inn i datastrukturen.
 */
void nyFugl()  {                                        /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2A - Leser KUN inn en fugl navn/art/type.
 *  @param   fugl  -  Fuglen det leses inn navnet til
 *  @param   nvn   -  Fuglens allerede innleste og memoryallokerte vnavn
 */
void fuglSettData(struct Fugl* fugl, char* nvn)  {      /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2A - Skriver dato (på formen MMDD) som DD/MM.
 *  @param   dato  -   Datoen som skal skrives ut som DD/MM
 */
void skrivDato(const int dato)  {                       /*  LAG INNMATEN  */    }
```

```
/**
 *  Oppgave 2B - Skriver ALLE fuglenes navn og antall observasjoner.
 */
void skrivAlleFugler()  {                                   /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2B - Skriver navn og antall observasjoner for EN fugl.
 *  @param  fugl  -  Fuglen det skrives ut navn og antall observasjoner om
 */
void fuglSkrivNoe(const struct Fugl* fugl)  {               /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2C - Skriver ALT om EN gitt fugl.
 */
void skrivEnFugl()  {                                       /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2C - Skriver ALT om en fugl.
 *  @param  fugl  -  Fuglen det skrives ut ALT om
 */
void fuglSkrivAlt(const struct Fugl* fugl)  {               /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2D - Registrerer (om mulig) EN ny observasjon av EN fugl.
 */
void nyObservasjon()  {                                     /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2D - Registrerer (om mulig) EN ny observasjon av fuglen.
 *  @param  fugl  -  Fuglen det registreres EN ny observasjon av
 */
void fuglNyObservasjon(struct Fugl* fugl)  {                /*  LAG INNMATEN  */      }

/**
 *  Oppgave 2E - Finner og skriver datoen med flest fugleobservasjoner.
 */
void dagenMedFlestObservasjoner()  {                        /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2F - Skriver HELE datastrukturen (ALLE fuglene) til fil.
 */
void skrivTilFil()  {                                       /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2F - Skriver ALT om EN fugl til fil.
 *  @param   ut    -  Filen det skal skrives ut til
 *  @param   fugl  -  Fuglen som får skrevet ut sine data
 */
void fuglSkrivTilFil(FILE* ut, const struct Fugl* fugl)  {  /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2G - Leser ALLE fuglene inn fra fil.
 */
void lesFraFil() {                                          /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2G - Leser ALT om EN fugl inn fra fil.
 *  @param  inn   -  Filen det skal leses inn fra
 *  @param  fugl  -  Fuglen som får innlest sine data
 *  @param  nvn   -  Fuglens navn som hittil er lest inn fra fil
 */
void fuglLesFraFil(FILE* inn, struct Fugl* fugl, const char* nvn)  {
                                                            /*  LAG INNMATEN  */    }
```