NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i PROG1001 – Grunnleggende programmering

Faglig kontakt under eksamen: **Frode Haug**

Eksamensdato: **17.desember 2024**

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>

char txt[]   = "Brokke-Valle-Dalen-Morgedal-Seljord-Notodden-Kongsberg";
char txt2[30] = "Norge";


int main() {
  char *t = txt, *t2;
  int   n = 1;

  printf("%i\n", !strncmp(txt+20, txt2+1, 4));

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

  t = txt;
  while (*t != '\0')  {
    if (*t == 'o') printf("%c ", *(t+1));  ++t;
  }
  printf("\n");

  t = txt+3;
  while ((t = strstr(t, "al")) != NULL)  {
    printf("%c ", *(t+3));  t += 2;
  }
  printf("\n");

  n = 0;   t = txt+19;   t2 = txt+31;
  while (*t != *t2)  {
    txt2[n++] = *t++;  txt2[n++] = *t2--;
  }
  txt2[n] = '\0';
  printf("%s\n", txt2);

  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 txt[][9] = { "Brokke", "Valle", "Dalen", "Morgedal" };
int  moh[]    = { 558, 312, 75, 433 };

struct Sted  {
  char nvn[20];
  int  moh;
};

struct Sted steder[4];

void H24Funk1()  {
  for (int i = 0;  i < 4;  i++) {
      strcpy(steder[i].nvn, txt[i]);   steder[i].moh = moh[i];
  }
}

void H24Funk2(const struct Sted s)  {
  printf("%s-%i  ", s.nvn, s.moh);
}

void H24Funk3(struct Sted* s1, struct Sted* s2)  {
  struct Sted s = *s1;  *s1 = *s2;  *s2 = s;
}

void H24Funk4(const char* t)  {
  for (int i = 0;  i < 4;  i++)
      if (strstr(steder[i].nvn, t)) printf("T "); else printf("A ");
}

bool H24Funk5(const int n, const int nn)  {
  return (steder[n].moh > nn);
}

char H24Funk6(const char* c)  {
 return ((char)(*c+2));
}

int main()  {

  H24Funk1();  H24Funk2(steder[3]);  H24Funk2(steder[0]);  printf("\n");

  H24Funk3(&steder[2], &steder[1]);  H24Funk3(&steder[1], &steder[0]);
               H24Funk2(steder[1]);  H24Funk2(steder[2]);  printf("\n");

  H24Funk1();  H24Funk4("al");  printf("\n");

  printf("%i %i %i\n", H24Funk5(3, 400), H24Funk5(1, 400), H24Funk5(0, 400));

  for (char* t = steder[3].nvn;  *t;  t++) printf("%c ", H24Funk6(t));
  printf("\n");

  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`, structene med datamedlemmer, (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.

Det skal holdes orden på ulike havner og (båt)rutene som går innom de ulike havnene.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayene `gHavner` og `gRuter`. I disse er indeksene fra 0 til `gAntallHavner/Ruter-1` i bruk. Vedlegget angir også hvilke datamedlemmer structene inneholder.
*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 skrivAlleHavner()` og `void havnSkrivData(const struct Havn* havn)`

Den første funksjonen sørger for at det blir gått igjennom alle aktuelle havner.
Den skriver havnens nummer (fra 1 og oppover), samt får (vha. den andre funksjonen) skrevet ut (*en* linje pr.havn) havnens navn og *om det er ulikt havnenavnet*: øyas navn.
NB: Legg merke til kommentaren i structen `Havn` om dette.

b) 4 Skriv innmaten til `void skrivAlleRuter()` og `void ruteSkrivData(const struct Rute* rute)`

Den første funksjonen gjør det samme som den første i oppgave 2a, når det gjelder ruter.
Den andre funksjonen skriver bare (på *en* linje) rutens (båt)navn *og kun* antall havner den er innom.

c) 12 Skriv innmaten til `void skrivAltOmEnRute()` og `void ruteSkrivAlt(const struct Rute* rute)`

Den første funksjonen kommer med en egen melding om det tomt med ruter. I motsatt fall spørres det om et aktuelt rutenummer (fra 1 og oppover). Deretter skrives *alt* om denne ruten ut hva den andre funksjonen. Denne funksjonen sørger for at følgende skrives ut: rutens (båt)navn, antall havner den er innom, navnene (og evt. øyene) på disse havnene, hvilke ukedager ruten er innom havnene (jfr. global `char`-array), pluss de tre datamedlemmene i `Rute` som hittil ikke er nevnt.
NB: Husk på å utnytte/bruke allerede lagde funksjoner (i oppgave 2a og 2b).

d) **12 Skriv innmaten til** `void nyHavn()` **og**
`void havnLesData(struct Havn* havn, char* nvn)`

Den første funksjonen kommer med en egen melding om det fullt med havner. I motsatt fall spørres det etter et nytt havnenavn. Finnes dette allerede, kommer det også en egen melding.
Ellers opprettes og legges det inn en ny havn. I den forbindelse (i den andre funksjonen) lagres det allerede innleste havnenavnet. Funksjonen leser også inn øynavnet som havna ligger på. Svarer brukeren *kun* ENTER, betyr det at disse navnene er like. Dermed lagres intet øynavn, men aktuell peker nullstilles bare. **NB:** Husk evt. å frigi allokert memory (om dette er gjort) ifm. at noen navn (havn og/eller øy) altså ikke skal lagres allikevel.

e) **12 Skriv innmaten til** `void nyRute()` **og** `void ruteLesData(struct Rute* rute)`

Den første funksjonen kommer med en egen melding om det fullt med ruter. I motsatt fall opprettes og legges det inn en ny rute, og alle dens data leses inn vha. den andre funksjonen. Som først spør om og leser inn *alle* structens enkeltvariable (som *ikke* er arrayer). Velg selv passende intervaller for `int`-verdiene. Deretter leser den inn alle (*unntatt* den siste) havnenavnene ruten/båten er innom. Den looper med dette inntil den har fått nok lovlige/eksisterende navn (deres indeks legges inn i aktuell array). Det aller siste navnet settes automatisk til å være det første (fordi *alle* ruter *alltid* ender opp igjen i starthavnen). Til slutt leses det inn hvilke ukedager ruten går/kjøres.
NB: Du trenger *ikke* å sjekke om båtnavnet også kjører andre ruter (for det kan det jo hende at den gjør), eller at den er innom duplikate havner (for det gjør den svært ofte på returen).

f) **12 Skriv innmaten til** `void mestTravleHavn()` **og**
`void ruteTellAnlop(const struct Rute* rute, int antall[])`

Funksjonene skal til sammen finne og skrive ut hvilke(n) havn(er) som har aller flest båtanløp i løpet av en *hel* uke. Starthavnen for *alle* ruter regnes *ikke* med, da den også telles i det båten kommer til sin siste havn (dagen før). `antall` bør være en lokal array inni `mestTravleHavn` som den sender fortløpende med til alle kallene av `ruteTellAnlop`.
NB: Alle ruter kjøres maksimum. en gang i døgnet.

g) **10 Skriv innmaten til** `void lesRuterFraFil()` **og**
`void ruteLesFraFil(FILE* inn, struct Rute* rute)`

Funksjonene sørger til sammen for at *alle* rutene blir lest inn fra filen «RUTER.DTA».
Filformatet bestemmer du helt selv, men ***skal* oppgis som en del av besvarelsen**.
NB: Alt om havnene skal ligge på en annen fil, men det skal det *ikke* lages kode for her.

Annet (klargjørende):

- Ordene «rute» og «båtrute» brukes synonymt i all teksten ovenfor og i koden (vedlegget).
- 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 neste sommers øyhopping i Hellas!
FrodeH

Vedlegg til PROG1001, desember 2024: Halvferdig programkode

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

#define   MAXHAVNER       100   ///<  Max. antall ulike havner.
#define   MAXRUTER        100   ///<  Max. antall ulike båtruter.
#define   MAXINNOM         20   ///<  Max. antall havneanløp på EN båtrute.
#define   STRLEN           80   ///<  Max. tekstlengde.
#define   DAGER             7   ///<  Antall dager i en uke.

                                ///<  Ukedagenes navn.
char dagnavn[DAGER][5] = { "Man", "Tirs", "Ons", "Tors", "Fre", "Lor", "Son" };


/**
 *  Havn (med havnas navn, og evt. øynavnet den ligger på).
 */
struct Havn  {
  char* navn;                   //  Havnas navn.
  char* oy;                     //  Øynavnet havna ligger på - OM DETTE ER
};                              //   ULIKT HAVNENAVNET, ELLERS ER DEN 'NULL'.


/**
 *  Rute (med båtens navn, lengde, plass til antall biler og personer, antall
 *        havner ruten er innom og indeksen i 'gHavner' for disse havnene,
 *        samt hvilke ukedager ruten er innom disse havnene).
 */
struct Rute  {
  char* baatnavn;
  int   lengde, antBiler, antPersoner;
  int   antallHavner;
  int   havnene[MAXINNOM];
  bool  dag[DAGER];
};


int  finnHavn(const char* nvn);                                 //  Ferdig-
void skrivMeny();                                               //   laget.
void skrivAlleHavner();                                         //  Oppgave 2A
void havnSkrivData(const struct Havn* havn);                    //  Oppgave 2A
void skrivAlleRuter();                                          //  Oppgave 2B
void ruteSkrivData(const struct Rute* rute);                    //  Oppgave 2B
void skrivAltOmEnRute();                                        //  Oppgave 2C
void ruteSkrivAlt(const struct Rute* rute);                     //  Oppgave 2C
void nyHavn();                                                  //  Oppgave 2D
void havnLesData(struct Havn* havn, char* nvn);                 //  Oppgave 2D
void nyRute();                                                  //  Oppgave 2E
void ruteLesData(struct Rute* rute);                            //  Oppgave 2E
void mestTravleHavn();                                          //  Oppgave 2F
void ruteTellAnlop(const struct Rute* rute, int antall[]);      //  Oppgave 2F
void lesRuterFraFil();                                          //  Oppgave 2G
void ruteLesFraFil(FILE* inn, struct Rute* rute);               //  Oppgave 2G


int    gAntallHavner = 0,           ///<  Antall havner hittil registrert.
       gAntallRuter  = 0;           ///<  Antall båtruter hittil registrert.
struct Havn* gHavner[MAXHAVNER];    ///<  Alle havner i registeret.
struct Rute* gRuter[MAXRUTER];      ///<  Alle havner i registeret.
```

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

  lesRuterFraFil();                                        //  Oppgave 2G
//  lesHavnerFraFil();                                     //  Skal IKKE lages.

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

  while (kommando != 'Q') {
    switch (kommando) {
      case 'H':  skrivAlleHavner();          break;         //  Oppgave 2A
      case 'R':  skrivAlleRuter();           break;         //  Oppgave 2B
      case 'E':  skrivAltOmEnRute();         break;         //  Oppgave 2C
      case 'N':  nyHavn();                   break;         //  Oppgave 2D
      case 'U':  nyRute();                   break;         //  Oppgave 2E
      case 'T':  mestTravleHavn();           break;         //  Oppgave 2F
      default:   skrivMeny();                break;
    }
    kommando = lesChar("\nValg");
  }

  return 0;
}



/**
 *  Prøver å finne indeksen i 'gHavner' for en navngitt havn.
 *
 *  @param   nvn  - Søkt havns navn
 *  @return  Indeksen for havnen, evt. -1 om ikke ble funnet
 */
int finnHavn(const char* nvn)  {
  for (int i = 0;  i < gAntallHavner;  i++)
      if (!strcmp(gHavner[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("\tH   = skriv alle Havner\n");
  printf("\tR   = skriv alle Ruter\n");
  printf("\tE   = skriv En rute\n");
  printf("\tN   = Ny havn\n");
  printf("\tU   = ny rUte\n");
  printf("\tT   = mest Travle havn\n");
  printf("\tQ   = Quit/avslutt\n");
}



/**
 *  Oppgave 2A - Skriver ALLE havnene (og om ulikt: øynavnet).
 *
 *  @see   havnSkrivData(...)
 */
void skrivAlleHavner()  {                          /*    LAG INNMATEN      */  }
```

```
/**
 *  Oppgave 2A - Skriver havnenavn og om ulikt: øynavnet.
 *
 *  @param  havn  -  Havnen det skrives ut noen data for
 */
void havnSkrivData(const struct Havn* havn)  {       /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2B - Skriver ALLE rutene.
 *
 *  @see   ruteSkrivData(...)
 */
void skrivAlleRuter()  {                             /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2B - Skriver båtnavn og antall havner ruten er innom.
 *
 *  @param  rute  -  Ruten det skrives ut noen data for
 */
void ruteSkrivData(const struct Rute* rute) {        /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2C - Skriver ALT om EN gitt rute.
 *
 *  @see   ruteSkrivAlt(...)
 */
void skrivAltOmEnRute()  {                           /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2C - Skriver ALT om en rute.
 *
 *  @param  rute  -  Ruten det skrives ut ALT om
 *  @see    ruteSkrivData(...)
 *  @see    havnSkrivData(...)
 */
void ruteSkrivAlt(const struct Rute* rute)  {        /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2D - Legger (om mulig) en ny havn inn i datastrukturen.
 *
 *  @see   havnLesData(...)
 */
void nyHavn()  {                                     /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2D - Leser inn ALT om EN havn.
 *
 *  @param   havn  -  Havnen det leses inn alle data for
 *  @param   nvn   -  Havnens allerede innleste og memoryallokerte navn
 */
void havnLesData(struct Havn* havn, char* nvn)  {    /*    LAG INNMATEN      */  }
```

```
/**
 *  Oppgave 2E - Legger (om mulig) en ny rute inn i datastrukturen.
 *
 *  @see   ruteLesData(...)
 */
void nyRute()  {                                    /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2E - Leser inn ALT om EN rute.
 *
 *  @param   rute  -  Ruten det leses inn alle data for
 */
void ruteLesData(struct Rute* rute)  {              /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2F - Finner og skriver havnen(e) med flest båtanløp.
 *
 *  @see   ruteTellAnlop(...)
 *  @see   havnSkrivData(...)
 */
void mestTravleHavn()  {                            /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2F - Registrerer rutens antall anløp i dets ulike havner.
 *
 *  @param   rute    -  Ruten det skal telles opp havneanløp for
 *  @param   antall  -  Array med antall anlop i hver av havnene
 */
void ruteTellAnlop(const struct Rute* rute, int antall[]) {
                                                    /*    LAG INNMATEN      */  }


/**
 *  Oppgave 2G - Leser ALLE bårutene inn fra fil.
 *
 *  @see   ruteLesFraFil(...)
 */
void lesRuterFraFil()  {                            /*    LAG INNMATEN      */  }


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