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:	**24.november 2020 - HJEMME**
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):	**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%)

I hele oppgave 1, være nøye med å :

- skrive *hele* funksjonen, dvs. *både* headingen og *hele* innmaten
- bruke `const` (når dette er aktuelt) ifm. parametre
- kommentere funksjonen (hva den gjør, evt. parametre og evt. returverdi) etter Doxygen

a) Lag funksjonen `bool lesSkrivOgReturnerOmTegnErLike()`

Funksjonen ber brukeren om å skrive inn tre tegn (ikke nødvendigvis bare bokstaver). Disse leses inn (blank, tabulator og linjeskift ignoreres). Den skriver så ut tegnene i baklengs rekkefølge (ift. hvordan de ble lest inn), med en '-' (bindestrek) i mellom hvert tegn.
Til slutt returneres `true` om minst to av tegnene er like, ellers returneres `false`.
(**NB:** Store og små bokstaver skal telle ulikt.)

b) Lag funksjonen `int forekomstAvBokstaver(.....)`

Funksjonen får inn to `char`-arrayer (`tekst` og `bokstaver`), samt to ulike `int`'er som parametre (som angir antall tegn i de to `char`-arrayene). Funksjonen skal finne ut og returnerer *totalt antall* forekomster av enkelt-tegnene i `bokstaver` i `tekst`.
F.eks om `bokstaver` er «aerst» og `tekst` er «Epler og bAnaneR smakEr Som ofTesT megEt godT», så vil den returnere 20 (da det er totalt 20 stk. av bokstavene 'a/A', 'e/E', 'r/R', 's/S' og 't/T').

NB: Store og små bokstaver teller likt. Alle tegnene i `bokstaver` er ulike.
Det ligger *ikke* en '\0' til slutt i de to `char`-arrayene.

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2g) *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`/`const`, `enum`, structen med datamedlemmer, funksjoner, globale variable, `main()` og ferdiglagd funksjon. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`. Bruk *alt* dette *svært* aktivt.

Det holdes orden på ulike gjenstander som skal ryddes ut av et hus, og hvilken status hver enkelt har. Alle gjenstandene har et `navn` (ikke nødvendigvis unikt), `hvem` som skal ha den (personnavn, butikk, organisasjon, «søppeldynga», Finn.no, …) evt. bare «`---`» om det ennå ikke er bestemt. Hver har også en `status` for hva som skal gjøres med den (se `enum`-verdiene i vedlegget).

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gGjenstander`. I denne er indeksene fra 0 til `gAntallGjenstander-1` i bruk. Vedlegget angir også hvilke datamedlemmer `Gjenstand` inneholder. **NB:** Gjenstandene angis for/av brukeren (ved I/O) fra 1 til `gAntallGjenstander`, selv om de i arrayen ligger lagret fra indeks nr.0 til `gAntallGjenstander-1`.

*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) **Skriv innmaten til** `void nyGjenstand()` **og**
`void gjenstandLesData(struct Gjenstand* g)`

Den første funksjonen kommer med en melding om det ikke er plass til flere gjenstander. I motsatt fall skrives den nye gjenstandens nummer ut på skjermen, og en ny `Gjenstand` opprettes. Den andre funksjonen leser inn gjenstandens `navn`, `hvem` settes til «`---`», og `status` settes til `Ubestemt`.

b) **Skriv innmaten til** `void skrivAlleGjenstandene()` **og**
`void gjenstandSkrivData(const struct Gjenstand* g)`

Den første funksjonen sørger for at *alle* gjenstandene og deres numre blir skrevet ut. Vha. den andre funksjonen blir *alle* data inni hver `Gjenstand` skrevet ut. Alle `enum`-ene skrives ut med en tilsvarende tekst som selve verdien, med *store* bokstaver.

c) **Skriv innmaten til** `void skrivAlleIKategori()`, `char lesKategori()`,
`enum Kategori konverterTilKategori(const char tegn)` **og**
`enum Kategori gjenstandHentStatus(const struct Gjenstand* g)`

Den første funksjonen spør om en *lovlig* bokstav (vha. den andre funksjonen), og får denne så omgjort til en `Kategori` (vha. den tredje funksjonen). Så skrives *alle* gjenstandene som er i den kategorien (vha. den fjerde funksjonen får man tak i dette for hver gjenstand) ut på skjermen.

Den andre funksjonen leser, *sikrer* og returnerer *en* bokstav som er den unike forbokstaven ('U','F','H','L','S','G') i hver av `enum`-verdiene. Den tredje funksjonen gjør om og returnerer 'U' som `Ubestemt`, 'F' som `Fjernet` osv. Den fjerde funksjonen: se vedlegget.

d) **Skriv innmaten til** `void settHvem()` **og**
`void gjenstandSettHvem(struct Gjenstand* g)`
Den første funksjonen spør om et gjenstandsnummer, 0 (null) inkludert. Skrives '0' kommer det en melding om at ingenting skjer. I motsatt fall kalles den andre funksjonen. Denne igjen skriver først hvem som for øyeblikket skal ha gjenstanden. Deretter sletter den nåværende `hvem`, og setter denne til å være en ny innlest verdi. Til slutt skriver den ut *alle* gjenstandens data.

e) **Skriv innmaten til** `void endreKategori()` **og**
`void gjenstandEndreKategori(struct Gjenstand* g)`
Den første funksjonen spør om et gjenstandsnummer, 0 (null) inkludert. Skrives '0' kommer det en melding om at ingenting skjer. I motsatt fall kalles den andre funksjonen. Denne igjen skriver *alle* gjenstandens data *både aller først og helt til slutt* i funksjonen. Mellom dette sørger den for at gjenstandens `status` endres (vha. *flere tidligere lagde* funksjoner)

f) **Skriv innmaten til** `void skrivTilFil()` **og**
`void gjenstandSkrivTilFil(FILE* ut, const struct Gjenstand* g)`
Funksjonene sørger til sammen for at *hele* datastrukturen blir skrevet til «GJENSTANDER.DTA». `gAntallGjenstander` *skal* ligge aller først på filen.
Filformatet ellers bestemmer du helt selv, men ***skal* oppgis i besvarelsen**.

g) **Skriv innmaten til** `void lesFraFil()` **og**
`void gjenstandLesFraFil(FILE* inn, struct Gjenstand* g)`
Funksjonene sørger til sammen for at *hele* datastrukturen blir lest inn fra «GJENSTANDER.DTA», etter det formatet du selv bestemte i forrige deloppgave.

Annet (klargjørende):

- Ordene «kategori» og «status» brukes noe synonymt i denne oppgaveteksten.
- 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 includer brukt i hele høst er tilgjengelig.

Lykke til med ryddingen av huset/hybelen til jul!

FrodeH

Vedlegg til PROG1001, 24.nov. 2020: Halvferdig programkode

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


#define   MAXGJENSTANDER 300     ///<  Max. antall gjenstander.
const int STRLEN = 60;           ///<  Max. tekstlengde.



/**
 *  Kategori (ulike kategorier/statuser for en gjenstand).
 */
enum Kategori { Ubestemt, Fjernet, Hentes, Lagres, Selges, Gratis };



/**
 *  Gjenstand (med navn, hvem skal ha den og status for den).
 */
struct Gjenstand {
    char* navn;                  //  Gjenstandens navn/beskrivelse.
    char* hvem;                  //  Hvem skal ha den/hvor skal den sendes.
    enum Kategori status;        //  Status for gjenstanden.
};


void skrivMeny();
void nyGjenstand();                                              //  Oppgave 2A
void gjenstandLesData(struct Gjenstand* g);                      //  Oppgave 2A
void skrivAlleGjenstandene();                                    //  Oppgave 2B
void gjenstandSkrivData(const struct Gjenstand* g);              //  Oppgave 2B
void skrivAlleIKategori();                                       //  Oppgave 2C
char lesKategori();                                              //  Oppgave 2C
enum Kategori konverterTilKategori(const char tegn);             //  Oppgave 2C
enum Kategori gjenstandHentStatus(const struct Gjenstand* g);    //  Oppgave 2C
void settHvem();                                                 //  Oppgave 2D
void gjenstandSettHvem(struct Gjenstand* g);                     //  Oppgave 2D
void endreKategori();                                            //  Oppgave 2E
void gjenstandEndreKategori(struct Gjenstand* g);                //  Oppgave 2E
void skrivTilFil();                                              //  Oppgave 2F
void gjenstandSkrivTilFil(FILE* ut, const struct Gjenstand* g);  //  Oppgave 2F
void lesFraFil();                                                //  Oppgave 2G
void gjenstandLesFraFil(FILE* inn, struct Gjenstand* g);         //  Oppgave 2G


int    gAntallGjenstander;         ///<  Antall gjenstander hittil registrert.
struct Gjenstand* gGjenstander[MAXGJENSTANDER];       ///<  Alle gjenstandene.


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

    lesFraFil();                                                 //  Oppgave 2G
    skrivMeny();
    kommando = lesChar("\nØnske");
```

```
    while (kommando != 'Q') {
        switch (kommando) {
           case 'N': nyGjenstand();                 break;  //  Oppgave 2A
           case 'A': skrivAlleGjenstandene();       break;  //  Oppgave 2B
           case 'K': skrivAlleIKategori();          break;  //  Oppgave 2C
           case 'H': settHvem();                    break;  //  Oppgave 2D
           case 'E': endreKategori();               break;  //  Oppgave 2E
           default:  skrivMeny();                   break;
        }
        kommando = lesChar("\nØnske");
    }

    skrivTilFil();                                          //  Oppgave 2F
    return 0;
}


/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tN   = Ny gjenstand\n");
    printf("\tA   = skriv Alle gjenstandene\n");
    printf("\tK   = skriv gjenstandene i en Kategori\n");
    printf("\tH   = endre/sett til Hvem/Hvor en gjenstand skal\n");
    printf("\tE   = Endre/sett kategori-tilhørighet\n");
    printf("\tQ   = Quit/avslutt\n");
}



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


/**
 *  Oppgave 2A - Leser inn relevante data og nullstiller andre.
 *
 *  @param   g -  Gjenstanden som får innlest/initiert sine data
 */
void gjenstandLesData(struct Gjenstand* g) {            /*  LAG INNMATEN  */  }



/**
 *  Oppgave 2B - Skriver ALT om ALLE gjenstander.
 *
 *  @see   gjenstandSkrivData(...)
 */
void skrivAlleGjenstandene() {                          /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2B - Skriver ALT om EN gjenstand ut på skjermen.
 *
 *  @param   g - Gjenstanden som skrives ut
 */
void gjenstandSkrivData(const struct Gjenstand* g)  {   /*  LAG INNMATEN  */  }
```

```
/**
 *  Oppgave 2C - Skriver ALT om ALLE gjenstander med EN GITT status.
 *
 *  @see   lesKategori()
 *  @see   konverterTilKategori(...)
 *  @see   gjenstandHentStatus(...)
 *  @see   gjenstandSkrivData(...)
 */
void skrivAlleIKategori()  {                                  /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2C - Leser LOVLIG kategori ('U', 'F', 'H', 'L', 'S', 'G').
 *
 *  @return  Forbokstaven i en LOVLIG kategori
 */
char lesKategori()  {                                         /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2C - Gjør om en (lovlig) bokstav til kategori, returnerer dette.
 *
 *  @param    tegn - Bokstaven som skal gjøres om til en kategori/enum-verdi
 *  @return   Kategori/enum-verdi
 */
enum Kategori konverterTilKategori(const char tegn)  {        /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2C - Returnerer kun gjentandens status.
 *
 *  @param    g  -  Aktuell gjenstand å returnere 'status' for
 *  @return   Gjenstandens 'status'
 */
enum Kategori gjenstandHentStatus(const struct Gjenstand* g) { /* LAG INNMATEN */ }




/**
 *  Oppgave 2D - Endrer (kanskje) EN aktuell gjenstands 'hvem'.
 *
 *  @see   gjenstandSettHvem(...)
 */
void settHvem()  {                                            /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2D - Endrer en gjenstands til 'hvem/hvor' den skal.
 *
 *  @param   g  -  Aktuell gjenstand å endre 'hvem' for
 *  @see     gjenstandSkrivData(...)
 */
void gjenstandSettHvem(struct Gjenstand* g)  {                /*  LAG INNMATEN  */  }
```

```
/**
 *  Oppgave 2E - Endrer (kanskje) EN aktuell gjenstands kategori/status.
 *
 *  @see   gjenstandEndreKategori(...)
 */
void endreKategori()  {                                    /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2E - Endrer en gjenstands kategori/status.
 *
 *  @param   g  -  Aktuell gjenstand å endre status for
 *  @see     gjenstandSkrivData(...)
 *  @see     lesKategori()
 *  @see     konverterTilKategori(...)
 */
void gjenstandEndreKategori(struct Gjenstand* g)  {        /*  LAG INNMATEN  */  }




/**
 *  Oppgave 2F - Skriver ALLE gjenstandene til fil.
 *
 *  @see   gjenstandSkrivTilFil(...)
 */
void skrivTilFil() {                                       /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2F - Alle EN gjenstands data skrives ut på fil.

 *
 *  @param   ut  - Filen det skal skrives til
 *  @param   g   - Gjenstanden som skrives til fil
 */
void gjenstandSkrivTilFil(FILE* ut, const struct Gjenstand* g){/* LAG INNMATEN */ }




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


/**
 *  Oppgave 2G - Leser ALT om EN gjenstand fra fil.
 *
 *  @param  inn - Filen det skal leses fra
 *  @param  g   - Gjenstanden som får innlest sine data
 *  @see    konverterTilKategori(...)
 */
void gjenstandLesFraFil(FILE* inn, struct Gjenstand* g) {  /*  LAG INNMATEN  */  }
```