Hemtentamen i Algoritmer och Datastrukturer

Startdatum:	2018-11-29
Slutdatum:	2018-12-07, kl. 16.00
Totalt antal poäng:	60p
Poäng för att få G:	20p
Poäng för att få VG:	40p
Ansvarig lärare:	TomKi, 076–775 06 29, tomas.kindahl@molk.se

Anvisningar

- Redovisa genom att maila in en zip (tarboll eller dylikt) till tomas.kindahl@molk.se.

Anvisningar

- Uppgifterna löses individuellt av eleven själv, d.v.s. eleven kodar och verifierar funktionen själv.
- Man får be elevkamrater och andra om råd, men man måste programmera och fatta designbesluten själv! Kopiera inte varandras kod! Om en gör fel och andra kopierar denna kod, så får alla fel – det är extremt onödigt. Skriv själv, eller kopiera från nätet!
- TomKi hjälper till med allmänna frågor, men inte med implementation eller kodning! Google är din vän!
- Om du råkar har gammal kod som du vill använda, använd den!
- **Speciella villkor** kan ställas i uppgiften som anger vad som är tillåtet och inte tillåtet, **följ dessa villkor**, annars kan hela uppgiften bli underkänd!
- I övrigt får man hämta information varsomhelst ifrån, man får ta kodsnuttar från nätet, men då skall man ange källan, och koden måste anpassas så att uppgiften genomförs korrekt i enlighet med uppgiftens krav. Ärlighet ger en välvillig bedömning från TomKi: att surfa på nätet och använda kod är en vedertagen problemlösningsmetod.

Poängsättning

- Där uppgiften kräver dokumentation ger kommentarer poäng, kommentera alltså såsom uppgiften kräver!
- Ändra inte på interface-delen (h-filen) av ett API, om inte uppgiften säger att du skall ändra i den eller lägga till kommentarer. Att ändra en typ i ett API, t.ex. en returtyp, ger avdrag, och om det ändras så mycket att kodning och testning inte fungerar, blir uppgiften **helt underkänd**.

Tentamensuppgifter

Det finns en kodbas i form av en zipfil `hemtenta.zip` på Moodle som skall användas att utgå ifrån, när uppgifterna i hemtentamen genomförs. Ladda ner den och packa upp den på ett lämpligt ställe.

När du har kodat klart skall du göra en `make clean` i katalogen `hemtenta` och därefter zippa ihop och sända till TomKi.

1. **modularisera class fraction**

   I katalogen `fraction` finns en fil `testfract.cpp` som innehåller en klass `fraction` som längtar efter att läggas i en separat modul. Den saknar också några metoder som du kan få poäng på att implementera.

   Deluppgifter:

   a. (4p) Lägg klassen i en separat fil som heter `fraction.hpp` och **inget annat**. Inkludera `fraction.hpp` i filen `testfract.cpp`. Sätt `#include` guards i `fraction.hpp` som man skall enligt konstens alla regler. Om du inte kom ihåg vad `#include` guards var, googla! Testkompilera!

   b. (6p) Skapa en separatkompilerad modul genom att lägga moduldefinitionerna i en ny fil `fraction.cpp`, och stubba metoderna i `fraction.hpp`! Lägg till ett alternativ i `Makefile` så att modulen `fraction.o` separatkompileras med hjälp av kommandot

      ```
      g++ -c fraction.cpp
      ```

      Utvidga även `Makefile` så att modulen`fraction.o` länkas in i huvudprogrammet!

   c. (4p) Snygga till filen `fraction.hpp` genom att kommentera den. Skriv en kommentar på 1-2 rader som förklarar vad `class fraction` används till, och för varje **publik** metod skriv en kommentar på en rad som förklarar vad metoderna gör.

   d. (2p) Det saknas subtraktion i klassen. Skriv en subtraktions-operator och lägg till i API:t och i implementationen. Den skall ha huvudet:

      ```
      fraction operator-(fraction F);
      ```

   e. (2p) Lägg in testkod för subtraktion i funktionen `testOps` i `testfract.cpp`!

   f. (2p) Skriv även en divisions-operator.

   g. (2p) Lägg in testkod för division i funktionen `testOps` i `testfract.cpp`!

2. **utvidga API mystring**

   I katalogen `mystring` i zip-bollen finns det redan implementerat ett bibliotek `mystring` som lagrar strängar som länkade listor. I filen `test-mystring.c` ser du hur strängar av typen `mystr` skapas. Din uppgift blir att implementera tre ytterligare funktioner `mystr_len`, `mystr_dup` och `mystr_inverse`.

   Du **får inte** ändra något i filen `mystring.h`! Om du ändrar något så att biblioteket inte längre fungerar så får du noll poäng på hela uppgiften. Om du gör någon annan ändring så blir det poängavdrag.

Funktionerna skall implementeras i filen `mystring.c`. Där får du ändra ganska mycket, men det skall gå igenom kompilatorn. Du behöver egentligen bara skriva de tre nedersta funktionerna.

a. (5p) Funktionen `mystr_len` ska räkna ut längden på en sträng, så att längden på `"Kalle"` blir 5. Implementera `mystr_len`! Att ta längd på en tom sträng skall hanteras på rätt sätt. Rätt sätt står beskrivet i `mystring.h`. Funktionen skall gå igenom testerna som finns i filen `test_mystring.c`.

b. (7p) Funktionen `mystr_dup` duplicerar en hel sträng, så att om man senare ändrar i kopian, så påverkas inte originalet. Implementera `mystr_dup`! Att kopiera en tom sträng skall hanteras på rätt sätt. Rätt sätt står beskrivet i `mystring.h`. Funktionen skall gå igenom testerna som finns i filen `test_mystring.c`.

c. (10p) Funktionen `mystr_inverse` skapar en sträng i omvänd ordning utifrån original-strängen, så att om du har originaltexten `"Kalle"` så får du resultattexten `"ellaK"`. Implementera `mystr_inverse` så att den fungerar på rätt sätt. Rätt sätt står beskrivet i `mystring.h`. Funktionen skall gå igenom testerna som finns i filen `test_mystring.c`.

3. **sortering på Arduino**

I katalogen `shellsort` i zip-bollen finns en "tom" sketch förberedd för att implementera en sorteringsalgoritm. För jämförelse finns även `insertionsort` medlevererad i zip-bollen.

a. (16p) `shellsort.ino`:

I Arduino-filen `shellsort.ino` är uppgiften nu att **implementera shellsort**. Googla för att hitta bra kod att anpassa till Arduinokoden. Ange källan som en kommentar i koden! Du kan också använda gammal kod som du själv skrivit i något sammanhang, ange i så fall varifrån den kommer.

Sladda upp en fotoresistor enligt bild sist i denna PDF för att generera snabbt föränderliga analoga data som kan läsas av intensivt snabbt, andra lösningar kan också godtas – redovisa i så fall hur.

mølk Algoritmer och Datastrukturer IoT17

Sökning och sortering > uppgift 2

★ Vi sätter upp en fotoresistor för att registrera ljusvärden på Arduinon.