Programmering 1

Inlämningsuppgift 8: Stränghantering, headerfiler

I denna kurs bestäms 30% av kursvitsordet utgående från resultatet av inlämningsuppgifterna. Denna inlämningsuppgift bidrar med 1/9 av det totala poängantalet. Se kursens hemsidor i Moodle för mera detaljerade bedömningsgrunder.

Ladda upp dina lösningar som en zip-fil till inlämningsmappen i Moodle före genomgångstillfället, dvs. **senast tisdagen den 31.10 kl. 12.30**. Försenade inlämningar ger inga poäng. Efter genomgångstillfället görs modellösningar tillgängliga på Moodle.

Obs: I dessa uppgifter är det både tillåtet och rekommendabelt att utnyttja funktioner för sträng- och bokstavshantering från **`stdio.h`**, **`string.h`** samt **`ctype.h`**.

1. Anagramkontroll (100%)

Ett anagram är ett ord som kan skapas genom att kasta om bokstäverna i ett annat ord. Din uppgift är att skriva ett program som kontrollerar om ett givet ord är ett anagram av ett annat ord.

För att lösa uppgiften skall följande algoritm användas:
- Användaren matar in två strängar
- Oväsentliga tecken raderas (mellanslag, punkter, komman…) och alla stora bokstäver görs om till små bokstäver.
- Därefter kan den egentliga anagramkontrollen genomföras:

```
OM de två strängarna är lika långa
    FÖR varje bokstav i den första strängen
        Kontrollera om bokstaven finns i den andra strängen
        OM JA, radera bokstaven från den andra strängen
    OM den andra strängen nu är tom är de två strängarna anagram
```

Observera att denna algoritm är utvald för att ge mångsidiga övningsmöjligheter med strängar och funktioner, och inte nödvändigtvis är den mest effektiva eller naturliga för att kontrollera anagram. Ett problem är att algoritmen modifierar strängarna, så för att också ha kvar de ursprungliga inmatade strängarna måste algoritmen arbeta med kopior av dessa.

Några programkörningar kan se ut som följer:

```
Enter first string: Mr. Mojo Risin'
Enter second string: Jim Morrison
Mr. Mojo Risin' is an anagram of Jim Morrison

Enter first string: Election results
Enter second string: Lies - let's recount
Election results is an anagram of Lies - let's recount

Enter first string: True
Enter second string: False
True is not an anagram of False
```

```
Enter first string: abcd
Enter second string: abce
abcd is not an anagram of abce
```

Programmet skall delas upp i följande funktioner. Avgör själv hur funktionerna skall anropa varandra.

`read_string()` skall låta användaren mata in en sträng med hjälp av **`fgets`**. Funktionen skall automatiskt **radera ett eventuellt radbytestecken** i slutet av strängen, och också **se till att inputbuffern är tom efter funktionsanropet**. Observera att metoden för att åstadkomma detta gås igenom under laboration 8.
Parametrar: en sträng och antalet bokstäver som kan läsas in (längden på strängen)
Returvärde: inget, men efter funktionsanropet skall strängen innehålla den inlästa datan.

`remove_char()` skall radera tecknet på ett angivet index i en sträng.
Parametrar: En sträng och ett index
Returvärde: inget, men efter funktionsanropet skall strängen vara ett tecken kortare än tidigare.

`contains_char()` skall kontrollera om en sträng innehåller ett visst tecken
Parametrar: En sträng och ett tecken
Returvärde: Indexet för tecknet om det kunde hittas, eller -1 om tecknet inte kunde hittas.

`normalize()` skall ändra alla stora bokstäver till små bokstäver och ta bort oönskade tecken från en sträng.
Vilka de oönskade tecknen är skall definieras genom en lokal strängvariabel, t.ex. **`" .,;:!'-"`**
Parametrar: En sträng
Returvärde: Inget, men efter funktionsanropet skall strängen vara normaliserad

`check_anagram()` skall kontrollera om två strängar är varandras anagram.
Parametrar: Två icke-normaliserade strängar. Observera att dessa strängar **inte** får modifieras inne i funktionen.
Returvärde: **`true`** om strängarna är anagram, annars **`false`**.

`main()` skall läsa in två strängar från användaren, kontrollera om strängarna är anagram och skriva ut resultatet.

Programkoden skall dessutom vara uppdelad på minst tre filer:

`anagram.c` ska innehålla funktionerna **`main`** och **`check_anagram`**

`utils.c` ska innehålla funktionerna **`read_string`**, **`remove_char`**, **`contains_char`** och **`normalize`**

`utils.h` ska innehålla funktionsdeklarationer för funktionerna i **`utils.c`**