En "bättre" strängimplementation för C

> Notera att denna uppgift är "klurig" så tillvida att specifikationen innehåller visst utrymme för tolkning och att det finns och/eller behov av dechiffrering.

Filen `string.h` i C's standardbibliotek innehåller funktioner för att manipulera textsträngar. C saknar en särskild strängtyp och representerar istället strängar som en `char`-array med ett `\0`-tecken (nulltecken) som sista tecken. En sträng kan skapas på följande sätt:

```
1  char *text = "spam spam spam"; // Creates a string
```

Att representera strängar på ett så enkelt sätt har vissa nackdelar. T.ex. har strängen ingen kunskap om sin egen längd, utan den måste räknas ut varje gång man vill t.ex. skapa en ny kopia av strängen, etc.

Det brukar sägas att varje inbiten C-programmerare tids nog kommer att implementera sitt eget strängbibliotek. Denna inlämningsuppgift går ut på att skriva ett förbättrat och plattformsoberoende strängbibliotek kallat `istring`. Biblioteket skall vara bakåtkompatibelt, dvs. strängar skapade med `istring` skall kunna användas som argument till funktioner som opererar på vanliga nullterminerade C-strängar. Däremot skall alla `istring`-funktioner förvänta sig `istring`-stränger som indata[1] och använda sig av

Biblioteket `istring` skall implementera den header-fil som finns i repositoryt. En `istring`-sträng är en `char`-array där de första fyra tecknen används för att lagra strängens längd som en `int`. Konceptuellt, givet att **`sizeof`**`(int) = 4`, kan vi se på strängen som denna typ (vilket inte är en legal C-typ (varför?!)):

```
1  typedef struct _istring
2  {
3    int length;
4    char str[length];
5  } _istring, *istring;
```

> Notera att header-filen för `istring`-biblioteket inte skall lämna ut någon information om hur en `istring` är internt representerad *eller* att det ens finns en **`struct`** för `istrings`. Hur fungerar det med testning? Vad är den enkla lösningen?

Bakåtkompatibiliteten bevaras genom att pekare till `istring`-strängar pekar fyra[2] tecken in i strängen, dvs. förbi `int`:en som sparar storleken och på första tecknet av den nullterminerade strängen. Detta skall i föreliggande fall lösas med en kombination av *pekararitmetik* och typomvandling. Pekararitmetik är särskilt felbenägen kod eftersom man "hoppar runt" i minnet genom att addera och subtrahera värden från minnesadresser. För att slippa skriva kod av typen `p-4` där `p` är en minnesadress definierar vi två hjälpmakron, `START(p)` och `STRING(p)`. Det första omvandlar en `char*`-pekare till en pekare till ("starten av") en `istring` och `STRING` gör det omvända. All pekararitmetik i modulen skall vara inkapslad i dessa makron.

C använder sig av ett s.k. globalt namespace, d.v.s alla namn som en modul exporterar bor i samma namnrymd, och det är inte möjligt att ha flera variabler eller funktioner med samma namn. Om vi vill kunna använda `string.h` och `istring.h` samtidigt måste alltså modulerna exportera olika namn. Funktionerna i `istring.h` har därför samma namn som i `string.h` men med ett inledande `i`.

Biblioteket ska ha följande egenskaper:

- Full bakåtkompabilitet med `string.h`, dvs. alla strängar skapade med den nya koden ska fungera med `string.h`.
- `istring`-biblioteket skall betrakta strängar som oföränderliga objekt, dvs. funktioner som t.ex. skall förändra strängar skall skapa och returnera en förändrad kopia. (Med undantag från `istrfixlen` och `istrslen`.)

En *inkomplett* uppsättning enhetstester och en (komplett) headerfil finns att checka ut. De funktioner som finns med i header-filen skall implementeras med samma funktionalitet som "motsvarande" funktion i `string.h`. Alltså, funktionen `istrcpy` skall fungera som `strcpy` i `string.h`, men operera på `istring`-strängar.

[1] Med några undantag, se header-filen.
[2] För något värde av 4 – d.v.s. vad som egentligen avses här är **`sizeof`**`(int)`.

Uppgiften examineras huvudsakligen genom att enhetstesterna körs. Detta kan göras genom att köra kommandot `make test`. Testerna för `istrslen` och `istrfixlen` finns inte utan de måste du skriva själv från grunden. För `istrcpy` och fyra andra funktioner finns testskelett som också måste implementeras.

> Ett strängbibliotek måste vara *plattformsoberoende!* Detta prövar vi här genom att kräva att testerna skall fungera *både* på institutionens SPARC- och X86-maskiner. Sidan `http://www.it.uu.se/datordrift/faq/unixinloggning` listar tillgänliga servrar och deras maskinvara.