Huiswerkopdrachten Programmeren in C – Week 3

De inleidende ***én*** uitleidende instructie bij de opdrachten van week 1 blijft ook voor deze week van kracht!

1. Maak de laatste lesopdracht af uit de sheets, hieronder een iets uitgebreider omschrijving:
   - Zoek uit hoe je de functie 'scanf()' gebruikt om gebruikersinput van de console in te lezen.
   - Merk op dat je een arraysize van voldoende grootte neemt en dat je input dus altijd kleiner is dan de totale array!
   - Schrijf een functie die een 'c-string' *by reference* accepteert en deze omgekeerd ***via zijn argument*** retourneert (return type is dus *void*).
   - Onderzoek in de functie eerst hoe lang de string is (tot char '\0'). De lengte van de array zelf is veel langer, maar het gaat natuurlijk om de lengte van de ingegeven string binnen die char-array. Let op dat verschil! *Het is wellicht een idee om hier een aparte sub-functie voor te schrijven*!
   - Tenslotte ***in place*** omdraaien! Je mag dus niet nóg een array reserveren om het resultaat naartoe te schrijven. Dit kan ook niet, omdat je het resultaat ook weer via het argument (by ref) retourneert. Uiteraard zijn tijdelijke chars om dingen even in op te slaan tijdens het swappen een goed idee ☺

   Hier een opzet voor je main:

   ```
   #define MAX_STR_LEN     100

   int main()
   {
           char a[MAX_STR_LEN];
           printf("Geef een string (max %d tekens): ", MAX_STR_LEN);
           scanf("%s", a);
           reverse(a);
           printf("%s\n", a);
           return 0;
   }

   void reverse(char *cstr) { … }
   ```

2. Voer bij een test met opdracht 1 een om te draaien string in waar een spatie in zit en let op wat er gebeurt. Zorg dat je deze tekortkoming repareert! Hint: heb je echt goed uitgezocht hoe 'scanf()' werkt, of heb je aangenomen dat mijn voorbeeld goed genoeg was ☺ ? Er is nog een alternatief naast 'scanf()'. Deze functie heet 'fgets()'. Zorg dat je het programma ook hiermee aan de praat krijgt.

3. Zoek op wat een **palindroom** is. Schrijf en demonstreer een functie die van een opgegeven string controleert of het een palindroom is. Je zou dit probleem kunnen oplossen door eerst de string om te draaien en daarna met de oorspronkelijke string te vergelijken. Dit is echter een 'computationeel dure' optie, zowel in (geheugen-)ruimte als (CPU-)tijd. **De functie moet daarom werken met twee pointers, één die wijst naar het eerste karakter, de andere wijst naar het laatste.** Gebruik pointer arithmetic om de pointers naar elkaar toe te laten tellen en je vergelijking op deze manier 'in place' uit te voeren.

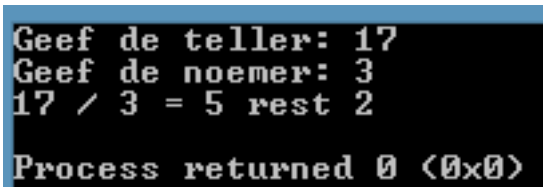
   Lees de string in vanaf de console middels een methode naar voorkeur, je hoeft hierbij geen rekening te houden met spaties (je mag de eenvoudige variant met 'scanf()' gebruiken). Je mag er verder van uitgaan dat de invoer altijd *lowercase* is. Gebruik de volgende declaratie voor je functie:

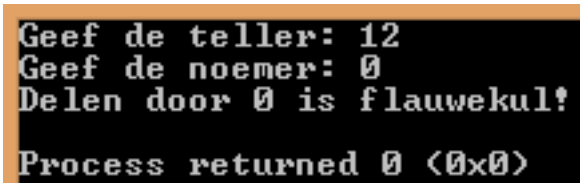
   ```
   bool is_palindrome(char *cstr) { … }
   ```

4. Schrijf en demonstreer een functie die een gehele deling uitvoert op twee natuurlijke getallen (unsigned int) en teruggeeft of het gelukt is (true) of dat men gepoogd heeft te delen door 0 (false). Invoercontrole (op negatieve getallen, kommagetallen of characters) mag je gemakshalve achterwege laten. Middels twee parameters moet de functie tevens het resultaat (= quotiënt) en de rest van de deling teruggeven (een variant van wat je in C# 'pass by output' zou noemen). Uiteraard implementeer je dat middels pointers. Gebruik de volgende signatuur voor je methode:

   bool deel_geheel(unsigned int teller, unsigned int noemer, unsigned int *quotient, unsigned int *rest)

   Je moet de volgende resultaten kunnen reproduceren:





   Probeer bij het testen de variabelen in je main() waarin je het quotiënt en de rest gaat opvangen eens aan te maken als 'losse pointers', waarna je ze meegeeft aan je functie, als volgt:

   unsigned int teller, noemer;
   unsigned int *quot, *rem;
   ... deel_geheel(teller, noemer, quot, rem) ...

   Waarom werkt dit niet? Hoe moet het wél?

5. Schrijf en demonstreer een functie ('max_value()') die de maximale waarde in een int-array zoekt door alléén gebruik te maken van pointers. **Hiertoe moet je pointer arithmetic en dereferentie goed door hebben**! Je mag in de functie één lokale variabele declareren om de maximale waarde in vast te houden om deze te retourneren als de functie 'klaar' is. De gewenste 'main()'-functie van het programma is hieronder gegeven in een screenshot. Zorg dat je begrijpt wat er in onderstaande code gebeurt!

```c
#include <stdio.h>
#include <stdlib.h>

#define ARR_SIZE   12

int max_value(int *some_array, int size);

int main (void)
{
    int i, max, the_array[ARR_SIZE];

    for (i = 0; i < ARR_SIZE; i++) *(the_array + i) = rand();
    for (i = 0; i < ARR_SIZE; i++) printf("%6d", *(the_array + i));
    printf("\n");

    max = max_value(the_array, ARR_SIZE);
    printf("Max = %6d\n", max);

    return 0;
}
```