Aufgabe A1

In dieser Aufgabe geht es um Palindrome. Definition: “Ein Palindrom ist eine Zeichenkette, die von vorn und von hinten gelesen gleich bleibt.” Beispiel: “otto”, oder “ein esel lese nie”. Die Funktion `is_palindrom` soll ermitteln, ob es sich bei dem übergebenen Text um ein Palindrom handelt.

a) Implementieren Sie die in c_A1.h angegebene Funktion
```
int is_palindrom(char * p)
```
und ermitteln Sie, ob es sich bei dem C-String `p` um ein Palindrom handelt. **[12 P.]**

b) Geben Sie eine 1 zurück, wenn es ein Palindrom ist, sonst eine 0. **[4 P.]**

c) Ändern Sie die Signatur der Funktion so ab, dass der C-String `p` in der Funktion nicht verändert werden darf. **[4 P.]**

Die Testfälle in `tc[]` in c_A1.c enthalten jeweils einen Text, der untersucht wird (`candidate`), sowie das erwartete Ergebnis der Untersuchung (`expected`).

Bewertungsschema			
A1	**Punkte maximal**	**Punkte erreicht**	**Kommentar**
a)	**12**		
b)	**4**		
c)	**4**		
	20		

Aufgabe A2 - Auswahlaufgabe

In dieser Aufgabe wird die Häufigkeit der einzelnen Zeichen (Histogramm) eines C-Strings ermittelt. Beispiel: In dem Text “hello” gibt es ein ‘e’, ein ‘h’, zwei ‘l’ und ein ‘o’.

Die Häufigkeit eines *einzelnen* Zeichens kann in dem extra dafür definierten Datentyp

```
typedef struct { char c; unsigned int n; } histogram_t;
```

abgelegt werden, und zwar das Zeichen in `c` und die Anzahl in `n`.

Die Funktion `calc_histogram` ermittelt nun die Häufigkeit *aller* vorkommenden Zeichen in einem Text und legt diese in einem Feld mit entsprechender Länge ab. Da die Anzahl der unterschiedlichen Zeichen zu Beginn nicht feststeht, wird durch die Funktion der Speicher für das Feld dynamisch angelegt. Das bedeutet für obigen Beispieltext “hello”, dass ein Feld von 4 `histogram_t` Strukturen dynamisch erzeugt wird, in das alphabetisch sortiert die Werte {‘e’,1} {‘h’,1} {‘l’,2} und {‘o’,1} abgelegt werden.

a) Implementieren Sie die Funktion
```
void calc_histogram(const char * text, histogram_t** p,
                    unsigned int * n)
```
und ermitteln Sie die Häufigkeitsverteilung der Zeichen im übergebenen C-String `text`. **[16 P.]**

b) Legen Sie den Speicher für das Feld der Häufigkeiten dynamisch an und geben den Zeiger auf das Feld über `p` zurück. **[8 P.]**

c) Legen Sie die Häufigkeiten der Zeichen alphabetisch sortiert ab (siehe oben). **[2 P.]**

d) Geben Sie die Anzahl unterschiedlicher Zeichen, also die Länge des Feldes aus b), über `n` zurück. **[2 P.]**

e) Besitzt der C-String `text` die Länge 0, besteht also aus keinem Zeichen, wird auch kein Speicher allokiert und NULL über `p` abgelegt sowie entsprechend 0 über `n` zurückgegeben. **[2 P.]**

Die Testfälle in `tc[]` in c_A2.c enthalten jeweils einen Text, der untersucht wird (`text`), sowie das erwartete Ergebnis der Untersuchung (`expected`) als Feld von `histogram_t` mit Länge (`count`).

Bewertungsschema			
A2	**Punkte maximal**	**Punkte erreicht**	**Kommentar**
a)	**16**		
b)	**8**		
c)	**2**		
d)	**2**		
e)	**2**		
	30		