FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Algorithmen & Datenstrukturen ILV
Rekursion

Abgabe: Details zu den Abgabemodalitäten finden Sie auf Moodle. Voraussetzung für die Bewertung ist, dass das Programm kompiliert und ausführbar ist. Die Lösungen werden jeweils am Anfang der nächsten (online) Übungseinheit von zufällig gewählten Studierenden vorgestellt.

Ziele:

- Schreiben von Funktionen
- Rekursion

Hinweise zum Autograding

Die Übungsaufgaben werden in ***recursion.cpp*** gelöst und in *main.cpp* getestet. Alle anderen Dateien bleiben unberührt! Zeile 2 in *main.cpp* - #define TEST 1 - bestimmt ob Tests ausgeführt werden, oder nicht. 0 bedeutet, dass nicht getestet wird. Für die Abgabe muss die Konstante immer den Wert 1 aufweisen.

In der *main* Funktion dürfen **keine eigenen return statements** gesetzt werden (K.O. Kriterium). Sonst werden keine Testfälle ausgeführt.

Entfernen Sie vor der automatischen Bewertung **sämtliche Eingabeaufforderungen (scanf, gets,...)** von der Konsole. Diese führen zu einem Timeout und verhindern das Autograding auf Github Classroom.

Übungsaufgabe 1

Harmonische Reihe
(5 Classroom Punkte)

Schreiben Sie ein Programm, dass eine **rekursive** Funktion implementiert, die der folgenden Definition entspricht:

$$H(n) = \begin{cases} 1 & \text{wenn } n = 1 \\ H(n-1) + (1/n) & \text{wenn } n > 1 \end{cases}$$

Die Funktion `float h(int n);` liefert für n > 0 die harmonischen Reihe als float Wert zurück. Ansonsten -1.

In *main()* könnte die Funktion *h(n)* dann mittels Eingabeaufforderung etwa wie folgt getestet werden:

```
n = 5
H(5) = 2.28
```

(Text in rot = Benutzereingabe)

Die Beschränkung auf zwei Nachkommastellen erfolgt in C mittels *printf("%.2f"...);*

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Übungsaufgabe 2

Palindrome
(5 Classroom Punkte)

Schreiben Sie eine **rekursive** Funktion, die überprüft ob es sich bei einem Wort oder Satz um ein Palindrom handelt. Der Prototyp sieht wie folgt aus:

```
int isPalindrom(char *text, int left, int right);
```

Hinweise: Der erste Parameter *text* zeigt auf den eingegebenen Text, *left* ist anfänglich der Index des ersten Zeichens im Text und *right* der Index des letzten Zeichens im Text. Die beiden Indizes bewegen sich im Laufe der Rekursion aufeinander zu. *left* wird erhöht, *right* wird verringert. Überlegen Sie hierfür eine geeignete Abbruchbedingung.

Es wird die Annahme getroffen, dass lediglich zusammengeschriebene Wörter in Kleinbuchstaben ohne Sonderzeichen als Text übergeben werden. Somit beschränken wir uns auf das wesentlichste - die Rekursion.

Beispiele zum selbständigen Testen in main():
(Text in rot = Benutzereingabe)

```
amanaplanacanalpanama
is a Palindrom.
```

```
helloworld
not a Palindrom.
```

```
a
is a Palindrom.
```

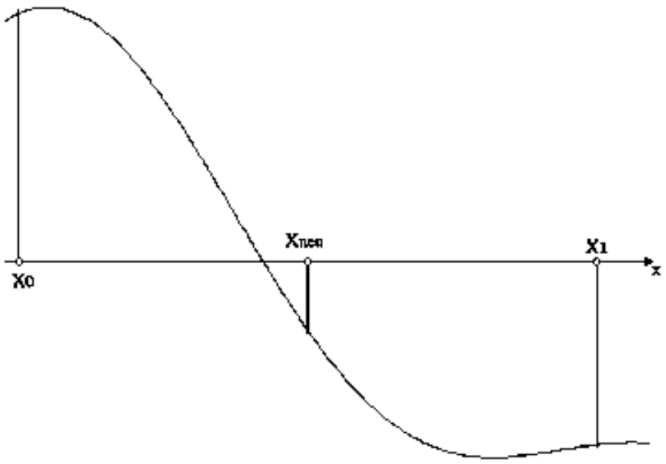
Übungsaufgabe 3

Nullstellensuche
(5 Classroom Punkte)

Schreiben Sie ein Programm zur Nullstellensuche einer mathematischen Funktion. Verwenden Sie dazu das Bisektionsverfahren, wie im Folgenden erläutert:

Dieses Verfahren basiert auf einer sehr einfachen Idee. Ein "fortlaufender" Funktionsgraph *y=f(x)*, dessen Wert *f(x0)* an der Stelle *x0* positiv und *f(x1)* an der Stelle *x1* negativ ist (oder umgekehrt), muss zwischen *x0* und *x1* die x- Achse schneiden. Durch wiederholtes Halbieren der Strecke *x0x1*, kann die Nullstelle Schritt für Schritt eingeschlossen werden. Der Mittelwert *xneu* zwischen *x0* und *x1* ist einfach zu berechnen:

$$\text{Mittelpunkt} \qquad xneu := \frac{x0 + x1}{2}$$



Implementieren Sie dieses Verfahren in einer **rekursiven** Funktion *bisection*. *x0*, *x1* und ein *epsilon* als Abbruchbedingung werden als *float* Paramater übergeben. Die Nullstellensuche soll solange ausgeführt werden, bis die Differenz von *x1-x0* um weniger als *epsilon* von Null abweicht. Die Funktion liefert die Nullstelle als *float* zurück. Der Prototyp sieht wie folgt aus:

```
float bisection(float x0, float x1, float epsilon);
```

Zusätzlich steht eine C-Funktion *f* zur Verfügung, die eine mathematischen Funktion abbildet. Die Funktion *bisection* greift auf diese zu, um Funktionswerte an den entsprechenden Stellen zu berechnen. Probieren Sie gerne unterschiedliche Funktionen zum Testen des Bisketionsverfahrens aus. Achten Sie vor der Abgabe darauf wieder auf die ursprüngliche, quadratische Funktion umzustellen.

Diskussion: Was passiert, wenn es im Intervall x0x1 keine Nullstelle gibt? Was könnten Sie tun, um diese Situation zu erkennen oder zu vermeiden?

HINWEIS ZUR BENOTUNG

Alle Aufgaben müssen rekursiv implementiert werden. Iterative Lösungen werden mit 0 Punkten bewertet.