Grundlagen der Praktischen Informatik **Sommersemester 2024**
Dr. Stefan Siemer Georg-August-Universität Göttingen
Timo Specht Institut für Informatik

Übungsblatt 11

Hinweis: Auf diesem Blatt können Sie ingseamt 70 **extra** Punkte sammeln. Dieses Blatt stellt eine kleine Wiederholung des Stoffes dar und bietet eine zusätzliche Programmieraufgabe.

E-Learning

Absolvieren Sie die Tests bis Di., 02.07., 14 Uhr

Sie können einen Test **nur einmal durchlaufen**. Sobald Sie einen Test starten steht Ihnen nur eine **begrenzte Zeit** zu Verfügung, um den Test zu bearbeiten.

Alle Punkte, die Sie beim Test erreichen, werden ihnen angerechnet.

Achtung

Manche Tests haben relativ lange Bearbeitungszeiten ($\geq$ 90 min.). Längere Inaktivitiät während dieser Zeit kann zum einem *timeout* und damit zu einem ungeplanten Ende des Tests führen.

Der Button *Test unterbrechen* verhindert den *timeout*, aber **stoppt nicht das weitere Ablaufen der Bearbeitungszeit**.

ILIAS – 45 Punkte

Verschiedenes

Absolvieren sie die Tests *GdPI 11 - ...*, die in der Stud.IP-Veranstaltung *Grundlagen der Praktischen Informatik (Informatik II)* unter *Lernmodule* hinterlegt sind.
(50 Punkte)

Praktische Übung

Testat Di., 02.07., ab 18 Uhr

Hilfe zum Bearbeiten der praktischen Übungen können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen. Die Testate finden ebenfalls in **Dreiergruppen** und Vierergruppen statt. Dabei sind die Gruppen identisch zu denen, die auch die theoretischen Aufgaben zusammen bearbeiten. In diesem Fall reserviert nur ein Gruppenmitglied einen Termin. Es ist ausreichend, wenn nur **eine** Person aus der Gruppe eine Prüfsumme abgibt.

Aufgabe 1 – 25 Punkte

Nicht-rekursiver Parser

Betrachten Sie folgende Parse-Tabelle, das Startsymbol ist 0.

	`id`	`(`	`)`	`:=`	`+`	`-`	`*`	`/`	`$$`
0	1								1
1	2 1								ε
2	`id := 3`								
3	5 4	5 4							
4	ε		ε		8 5 4	8 5 4			ε
5	7 6	7 6							
6	ε		ε		ε	ε	9 7 6	9 7 6	ε
7	`id`	`( 3 )`							
8					`+`	`-`			
9							`*`	`/`	

Erstellen Sie ein Python-Skript für einen nicht-rekursiver Parser, der die vorstehende Parse-Tabelle benutzt.

Allgemeine Anforderungen

1. Die Terminale werden als Strings, die Nichtterminale als ganze Zahlen codiert.
2. Kommentieren Sie Ihre Code.
3. Bei einem Fehler wird das Skript mit einer Fehlermeldung abgebrochen.

```
print("ERROR: ...")
exit()
```

4. Obwohl Python weitreichende Möglichkeiten zur Strukturierung und Fehlerbehandlung bietet, reicht für diese Übung ein einfaches Skript, das die Aufgabenstellung (gradlinig) abarbeitet.
5. Ein interaktives Einlesen der Eingabe ist nicht nötig.
6. Testen Sie Ihr Skript mit der Eingabe **`id:=id*(id+id)`** und einer Eingabe, die nicht zur Sprache gehört.

(5 Punkte)

Scannen der Eingabe

1. Die Eingabe ist ein String.
2. Die Menge der Terminale ist ein Tupel.
3. Mit Hilfe der Menge der Terminale wird aus der Eingabe eine Liste von Terminalen erzeugt, an die als letztes Element die Markierung für das Ende der Eingabe angehängt wird.

(5 Punkte)

Parsen der Terminalliste

1. Die Parse-Tabelle ist ein Tupel, das für jede Zeile der Parse-Tabelle ein Dictionary enthält.

   Die Dictionaries haben nur Einträge für die Zellen der Parse-Tabelle, in denen die rechte Seite einer Produktion steht.

   Die Schlüssel der Dictionaries sind Terminale, die Werte sind Tupel von Nichtterminalen und Terminalen. Ein leeres Tupel `()` repräsentiert ε.

2. Der Stapel des Parsers ist eine Liste von Terminalen und Nichtterminalen, die während des Parsens verändert, nicht kopiert, wird.

   Sie können auf ein Zeichen verzichten, das den leeren Stapel anzeigt.

3. Für das Parsen bietet sich eine **`while`**-Schleife an, die läuft, solange der Stapel nicht leer ist.

4. Eine Liste, die die Elemente einer Liste/eines Tupels **`xs`** in umgekehrter Reihenfolge enthält, kann man mit **`xs[::-1]`** erzeugen.

(15 Punkte)