Übung 06

Korrektur: Sebastian Pritz

Name: Marco Sarkady

**Punkte: 89**

Allgemeine Anmerkungen:

Für Testfälle sollten (wie in jeder Übung zuvor) ein eigenes Testfile erstellt werden -2

Bitte beim Projekt und Files umbenennen aufpassen. Prefix\_parser.cpp gab es nicht und hieß stattdessen „.cpp“ – also praktisch nur die Dateiendung...

Allgemein gute Abgabe, bis auf die paar Sachen beim letzten Punkt! 😊

# Beispiel 1: Arithmetische Ausdrücke (infix) 30 von 30

Lösungsidee: 8/8

|  |  |
| --- | --- |
| **Anmerkung** | **Abzug** |

Quellcode: 12/12

|  |  |
| --- | --- |
| **Anmerkung** | **Abzug** |

Testfälle: 10/10

|  |  |
| --- | --- |
| **Anmerkung**  Passt 😊 | **Abzug** |

# Beispiel 2: Arithmetische Ausdrücke (präfix) 30 von 30 Lösungsidee: 8/8

|  |  |
| --- | --- |
| **Anmerkung**  Der letzte Abschnitt gehört thematisch eig. zur Diskussion der Testfälle, aber sonst okay. Mehr dazu unten. | **Abzug** |

Quellcode: 12/12

|  |  |
| --- | --- |
| **Anmerkung**  Kein „großer“ Fehler, aber keine Ahnung wie das überhaupt kompiliert hat, wenn die using Statements über den imports sitzen. Bei mir tat es das auf jeden Fall nicht, da der Compiler ja nicht mal weiß, was „std::string“ überhaupt ist, bevor man es importiert. 🤔 | **Abzug** |

Testfälle: 10/10

|  |  |
| --- | --- |
| **Anmerkung**  Dass Infix-Ausdrücke nun nur die erste Zahl parsen ist der gegebenen Grammatik geschuldet, und nicht ein Implementierungsfehler per se. Eine Zahl ist ein Terminalsymbol 🡪 Wenn sie geparsed wird, wird nicht an andere Grammatikregeln geforwarded. Fehlermeldungen können also nur geworfen werden, wenn 1) ein Zeichen fehlt, oder 2) ein Zeichen nicht dem erwarteten entspricht.  Passt! 😊 | **Abzug**  **Hinweis** |

# Beispiel 3: Rechnen mit Variablen 31 von 40

Lösungsidee: 7/9

|  |  |
| --- | --- |
| **Anmerkung**  Keine Veranschaulichung der Grammatik zum Erlauben von Variablen | **Abzug**  -2 |

Quellcode: 18/18

|  |  |
| --- | --- |
| **Anmerkung** | **Abzug** |

Testfälle: 6/13

|  |  |
| --- | --- |
| **Anmerkung**  Keine komplexen Rechnungen mit Variablen (Klammern, verschachtelt...), mehrstelligen Zahlen, sowie kein Überprüfen ob die alten Rechnungen (komplett ohne Variablen) noch funktionieren.  Kein Test mit einer nicht definierten Variable  Kein Test mit „Error parsing factor“ 🡪 z.B zu viele Rechenzeichen | **Abzug**  **-3**  **-3**  **-1** |