Am Mittwoch, 3. Dezember 2014 lädt die Fachschaft Informatik ab 14 Uhr wieder alle Informatik-Studis zur Vollversammlung in HS 2, INF 288 ein. Weitere Informationen finden sich unter https://mathphys.fsk.uni-heidelberg.de/w/vv2014/

Beachten Sie die Hinweise zum Programmierstil auf Blatt 5. Bei groben Verstößen können Ihnen ab dieser Woche Punkte abgezogen werden.

ÜBUNG 6.1 UMGEBUNGEN

Gegeben sei das folgende Programm:

```
5  int g = 0;
6
7  int a_mod_b( int a, int b)
8  {
9      int m = a%b;
10     return m; [1]
11 }
12
13 int ggT( int a, int b)
14 {
15     g = g+1;
16     int Null=0;
17     if (b==Null)
18         return a; [2]
19     else
20         return ggT(b, a_mod_b(a,b));
21 }
22
23 int main ( void )
24 {
25     int a = 2;
26     int b = 14;
27     {
28         int a = 7;
29         int g = ggT(b,a);
30         b=g;
31     }
32     a=g;
33     return 0; [3]
34 }
```

Verwenden Sie das Umgebungsmodell (Definition 8.2 aus der Vorlesung) um zu bestimmen, welche Umgebungen existieren, wenn das Programm die mit [1], [2] und [3] markierten Zeilen erreicht. Die markierten Zeilen sollen erreicht sein, aber noch nicht ausgeführt – im Falle [3] ist Zeile 32 ausgeführt worden, Zeile 33 aber noch nicht.

Stellen Sie die Umgebungen graphisch dar (wie in der Vorlesung) um zu zeigen welche Namen existieren und welche Werte Sie haben. Namen ohne definierten Wert markieren Sie mit `?`.

6 Punkte

ÜBUNG 6.2 PRIMFAKTORZERLEGUNG

Schreiben Sie ein Programm, welches für eine gegebene Zahl deren Primfaktorzerlegung berechnet und diese auf der Konsole ausgibt. Schreiben Sie ihr Programm so, dass die Primfaktoren automatisch vom kleinsten zum größten sortiert ausgegeben werden. Beachten Sie außerdem die Bemerkung im Skript, daß man bei der Suche nach Teilern nur den Bereich $2, \ldots, \sqrt{n}$ absuchen muss. [*4 Punkte*]

Bestimmen Sie die algorithmische Komplexität Ihres Programms. [*2 Punkte*]

6 Punkte

ÜBUNG 6.3 EIN EINFACHER TASCHENRECHNER

Ziel dieser Aufgabe ist es, ein Programm zu schreiben, das eine Zeichenfolge als Eingabe erhält, versucht, diese als mathematische Formel in Umgekehrter Polnischer Notation (Postfixnotation)* zu interpretieren, und entweder das Ergebnis der Berechnung (gültiger Ausdruck) oder eine Fehlermeldung (ungültiger Ausdruck) ausgibt. So soll zum Beispiel die Zeichenfolge

*Siehe `http://de.wikipedia.org/wiki/Umgekehrte_Polnische_Notation` oder Übung 2.1

`67 55 - 54 6 / + 2 *`

zur Ausgabe “42” führen, denn es ist die Postfixnotation für die Formel

$$(((67 - 55) + (54/6)) \cdot 2).$$

Für die Eingabe sollen die folgenden Regeln gelten:

- Ziffernfolgen stellen Integer-Zahlen dar
- Die Symbole ’+’,’-’,’*’ und ’/’ stehen für die entsprechenden Operationen, wie Sie sie schon in C++ benutzen
- Alle weitere Zeichen werden ignoriert und haben nur den Effekt, zwei Ziffernfolgen und damit Zahlen voneinander zu trennen, falls sie zwischen diesen auftauchen

Benutzen Sie einen Stack, auf dem Sie der Reihe nach die gelesenen Zahlen ablegen. Dieser soll ein `struct` sein, das ein `int`-Array fester, ausreichender Größe und ein `int` zum Zählen der hinterlegten Elemente enthält.

Sie dürfen keine Funktionen verwenden, die Zeichenketten in Zahlen umwandeln, und müssen also “zu Fuß” die Zahlen Zeichen für Zeichen konstruieren. Benutzen Sie dafür ein `bool`, um sich zu merken, ob das vorherige Zeichen eine Ziffer war, und ein `int`, in dem Sie die Zahl zusammensetzen, bis die letzte Ziffer gelesen ist. Beachten Sie, dass das `push()` zum Speichern der Zahl auf dem Stack erst bei der nächsten Nichtziffer erfolgt, denn erst dann können Sie sicher sein, dass die Zahl zu Ende gelesen wurde.

Sobald einer der vier Operatoren gelesen wird, müssen Sie sich zwei Zahlen vom Stack holen, auf diese den Operator anwenden und das Ergebnis zurück auf den Stack legen. Auf diese Weise wird der binäre Baum der Formel (vgl. Übung 4) an den Blättern “gekürzt”, bis am Ende ein äquivalenter Baum mit nur einem Knoten (dem Ergebnis) übrig bleibt.

Auf der Homepage zur Vorlesung finden Sie die Datei `taschenrechner.cc`. Benutzen Sie diese als Vorlage für Ihr Programm. Testen Sie Ihr Programm mit den folgenden Eingaben:

- `"67 55 - 54 6 / + 2 *"`
- `"123456789987654321"`
- `"2 7 * - 4 +"`
- `"16 2 2 4 2 /*Dies ist (k?)ein Kommentar*/ 4 2 / *"`

Beachten Sie, dass die Anführungszeichen rund um das Kommandozeilenargument essentiell sind. Die Shell würde die Eingabe sonst aufgrund der enthaltenen Leerzeichen in mehrere Argumente zerlegen. *8 Punkte*