REVERSED POLISH NOTATION

Reversed polish notation, of RPN, is een manier om formules te noteren zonder haakjes.
Eerst geven we de getallen, dan geven we de operator.
Voorbeeld :

`7 2 +` is hetzelfde als: `(7 + 2)`

Alle basale rekenkundige operatoren zijn binaire operatoren, dwz ze werken met 2 getallen.
In deze opgave werken we **uitsluitend** met integers.
We kennen 4 operatoren : plus ('`+`') , min ('`-`'), maal ('`*`') en deel ('`/`').
Ook de deel-operator geeft een **int** retour ! (fractie wordt gewoon weggegooid)
Bij de **min** en **deel** is de volgorde van de getallen van belang ! Volg deze afspraak:

`9 6 -` is hetzelfde als `(9-6)`

en `15 7 /` is hetzelfde als `(int)(15 / 7)` (= 2)

Instructies :

RPN maakt op een 'natuurlijke' manier gebruik van een stack. Dit is een FILO lijst. (First In Last Out).
Een aangeboden string met de rpn formule moet eerst in afzonderlijke tokens verdeeld worden.
Een token is elke deelstring tussen spaties (of whitespaces om precies te zijn).
Elk token is een (string weergave van) een integer òf een operator.

1. Splits de rpn-string in tokens (deze fie is gegeven)
2. Is het token een integer dan wordt deze int op de stack geplaatst.
3. Is het token een operator dan worden de laatste twee int's van de stack gehaald. We laten de operator werken op de twee int's en plaatsen het resultaat van deze berekening op de stack.
4. Herhalen tot de gehele rpn-string is verwerkt (of totdat een fout is geconstateerd.)
5. Geef de uitkomst retour via een reference parameter.

Gegeven is de volgende code :

rpn.cpp:

```
#include <iostream>
#include <sstream>
#include <iterator>
#include "rpn.h"

… hier de functies uit rpn.h …

// deze functie krijg je gratis …
std::vector<std::string> getTokens(std::string line){
        std::istringstream iss(line);
        std::vector<std::string> tokens {std::istream_iterator<std::string>{iss},
                                         std::istream_iterator<std::string>{}};
        return tokens;
}
```

REVERSED POLISH NOTATION

```
int main()
{
    …  hier je 'test-code' b.v.: …
    std::string rpn_line = "12 3 4 * + 7 - 3 /";
    int uitkomst = 0;
    bool resultaat = compute( rpn_line, uitkomst);

    return 0;
}
```

Condities:

- Het programma moet natuurlijk met elke opgegeven string werken.

Merk op:

1. De inhoud van de '***main***' routine is niet van belang voor de test.
   Je mag hier test-code opnemen voor de gemaakte functies, geheel naar eigen inzicht.
2. De 'signatuur' van de functies mag niet worden aangepast. Wijzigingen geven fouten in de test.

Te uploaden bestanden:

1. rpn.cpp