問2　一般的な表記法の数式を逆ポーランド表記法に変換するアルゴリズムに関する次の記述を読んで，設問1～3に答えよ。

逆ポーランド表記法とは，演算子を二つの演算対象の後ろに配置することによって，数式を表現する表記法である。例えば，一般的な表記法の数式 1+2×3 を，逆ポーランド表記法では1 2 3 × + と表記する。

逆ポーランド表記法で表記した数式は，数値や演算子を左から順に処理すればよく，括弧を使う必要もないので，コンピュータが数式を取り扱うのに都合が良い。

一般的な表記法の数式から逆ポーランド表記法への変換は，スタックを用いることで容易に実現できる。

〔逆ポーランド表記法への変換アルゴリズム〕

一般的な表記法の数式を逆ポーランド表記法に変換するアルゴリズムでは，まず変換前の数式を，数値，演算子及び括弧の要素（以下，演算要素という）に分解する。演算子には二項演算子＋，－，×，÷を用い，括弧には“（”と“）”を用いる。数値は0～9の1桁の数とする。それぞれの演算要素には，優先度を定義する。

変換の処理には，変換前配列，スタック及び変換後配列を用いる。初期状態では，変換前配列には変換前の数式の演算要素が順に入っており，スタックと変換後配列は空の状態である。変換後には，変換後配列に逆ポーランド表記法に変換した結果が入る。例えば，変換前の数式が1+2×3の場合，初期状態は表1のようになる。

**表1　初期状態**

| 変換前配列 | スタック | 変換後配列 |
|---|---|---|
| 1 ＋ 2 × 3 | 空 | 空 |

逆ポーランド表記法への変換は，次の(1)～(4)の手順で行う。

(1)　変換前配列の先頭から順に，演算要素を1個参照する。
　参照する演算要素がない場合は(4)に進む。

(2)　スタック上に演算要素がある場合は，スタックの先頭にある演算要素の優先度を参照し，(1)の演算要素の優先度以上なら，スタックの先頭の演算要素をポップし，

変換後配列の末尾に追加する。これを，スタックの先頭の演算要素の優先度が，(1)の演算要素の優先度未満になるまで繰り返す。ただし，スタックの先頭の演算要素が“(”の場合は，そこで繰返しを終了する。

(3) (1)で参照した演算要素が“)”なら，それを破棄し，その際スタックの先頭にあるはずの“(”もポップして破棄した後(1)に戻る。(1)で参照した演算要素が“)”以外なら，その演算要素をスタックにプッシュし，(1)に戻る。

(4) スタック上にある全ての演算要素を順番にポップし，変換後配列の末尾に追加する。

演算要素の優先度を表2に，数式1+2×3を変換するときの処理過程を図1に示す。なお，図1中の丸で囲った演算要素は，(1)の手順で参照した演算要素である。

**表2　演算要素の優先度**

| 演算要素 | 優先度 |
|---|---|
| ( | 5 |
| 数値 | 4 |
| ア，イ | 3 |
| ウ，エ | 2 |
| ) | 1 |

注記　値が大きいほど優先度が高い。

処理の順序 ↓

| 手順 | 変換前配列 | スタック | 変換後配列 |
|---|---|---|---|
| (1) | (1) + 2 × 3 | | |
| (3) | (1) + 2 × 3 | 1 | |
| (1) | 1 (+) 2 × 3 | 1 | |
| (2) | 1 (+) 2 × 3 | | 1 |
| (3) | 1 (+) 2 × 3 | + | 1 |
| (1) | 1 + (2) × 3 | + | 1 |
| (3) | 1 + (2) × 3 | + 2 | 1 |
| (1) | 1 + 2 (×) 3 | + 2 | 1 |
| (2) | 1 + 2 (×) 3 | + | 1 2 |
| (3) | 1 + 2 (×) 3 | + × | 1 2 |
| (1) | 1 + 2 × (3) | + × | 1 2 |
| (3) | 1 + 2 × (3) | + × 3 | 1 2 |
| (4) | 1 + 2 × 3 | | 1 2 3 × + |

注記　スタックについては，右端の演算要素がスタックの先頭である。

**図1　数式1+2×3を変換するときの処理過程**

〔逆ポーランド表記法への変換プログラム〕

逆ポーランド表記法への変換プログラムを作成した。プログラム中で使用する主な変数，配列及び関数を表３に，作成したプログラムを図２に示す。

図 2 のプログラムでは，スタックの取扱いを容易にするために，ダミーの演算要素 null を定義し，プログラム開始直後にスタックにプッシュしている。

null がスタックからポップされることがないようにするために，その優先度を［　オ　］と定義する。こうすることで，プログラム中，手順(2)の処理を行う部分で［　カ　］の判定処理を記述する必要がなくなり，プログラムが簡潔になる。

**表３　図２のプログラム中で使用する主な変数，配列及び関数**

| 種別 | 名称 | 型又は戻り値の型 | 説明 |
|---|---|---|---|
| 関数 | GetElement ( index ) | 演算要素 | 変換前配列の index 番目の演算要素を返す。index が 1 の場合は先頭の演算要素を返す。 |
| 変数 | elementCount | 正の整数 | 変換前の数式に含まれる演算要素の個数。 |
| 関数 | GetPriority ( element ) | 非負の整数 | element で指定された演算要素の優先度を表す非負の整数を返す。 |
| 配列 | result[ ] | 演算要素の配列 | 変換後配列。添字は 1 から始まる。配列の大きさは十分に大きいものとする。 |
| 変数 | resultCount | 非負の整数 | 変換後配列にある演算要素の個数。 |
| 配列 | stack[ ] | 演算要素の配列 | スタックとして使用する配列。添字は 1 から始まる。配列の大きさは十分に大きいものとする。 |
| 変数 | stackCount | 非負の整数 | スタックにある演算要素の個数。 |

```
resultCount ← 0
stackCount ← 1
stack[stackCount] ← null
i ← 1
while (i が elementCount 以下の間繰り返す)
    elementPriority ← GetPriority(GetElement(i))
    stackTop ← [ キ ]
    // 演算要素をスタックからポップし，変換後配列の末尾に追加する。
    while ( [ ク ] かつ stackTop が "(" 以外の間繰り返す)
        resultCount ← resultCount + 1
        result[resultCount] ← stackTop
        stackCount ← stackCount − 1
        stackTop ← [ キ ]
    endwhile
    // 変換前配列を参照し，演算要素を処理する。
    if ( [ ケ ] )
        stackCount ← stackCount − 1
    else
        stackCount ← stackCount + 1
        stack[stackCount] ← GetElement(i)
    endif
    i ← i + 1
endwhile
// スタックに残った演算要素を，変換後配列の末尾に追加する。
while ( [ キ ] が null でない間繰り返す)
    resultCount ← resultCount + 1
    result[resultCount] ← [ キ ]
    stackCount ← stackCount − 1
endwhile
```

① ←

**図２　逆ポーランド表記法への変換のプログラム**

〔エラーチェックの追加〕

図２のプログラムの①の箇所において，i が２以上のとき，GetElement(i−1)の優先度と，GetElement(i)の優先度を比較することによって，簡易的な入力エラーチェックを追加することができる。例えば，数値の演算要素が 2 個以上連続する場合や，“ ) ” の直後に “ ( ” が続く場合など，四則演算の式として不正なものがあった場合はエラーとする。

入力エラーとする条件を表４に示す。GetElement(i−1)の優先度と，GetElement(i)

の優先度について，表 4 に従って評価をした結果，“OK”の場合は，そのまま処理を続行する。評価した結果が“Err”の場合は，入力された数式が誤っていると判断して処理を中断する。

表 4　入力エラーとする条件

| | | GetElement(i)の演算要素の優先度 | | | | |
|---|---|---|---|---|---|---|
| | | 5（“(”） | 4（数値） | 3(ア, イ) | 2(ウ, エ) | 1（“)”） |
| GetElement(i−1)の演算要素の優先度 | 5（“(”） | OK | OK | Err | Err | Err |
| | 4（数値） | Err | コ | サ | サ | OK |
| | 3(ア, イ) | OK | シ | ス | ス | Err |
| | 2(ウ, エ) | OK | シ | ス | ス | Err |
| | 1（“)”） | Err | Err | OK | OK | OK |

**設問 1**　逆ポーランド表記法への変換について，(1)，(2)に答えよ。

(1)　数式(2+3)×4 を逆ポーランド表記法に変換した結果を答えよ。

(2)　表 2 及び表 4 中の［ア］～［エ］に入れる適切な二項演算子を，＋，－，×，÷の中から一つずつ選んで，表を完成させよ。

**設問 2**　〔逆ポーランド表記法への変換プログラム〕について，(1)～(3)に答えよ。

(1)　本文中の［オ］に入れる適切な数値を答えよ。

(2)　本文中の［カ］に入れる適切な字句を 20 字以内で答えよ。

(3)　図 2 中の［キ］～［ケ］に入れる適切な字句を答えよ。

**設問 3**　表 4 中の［コ］～［ス］に入れる適切な字句を答えよ。