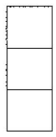
〔Ⅲ〕 次の説明を読み，下の問に答えなさい。

プログラミングにおけるデータの格納方法として「スタック」がある。スタックのイメージは，次の図のような入れ物に対して，



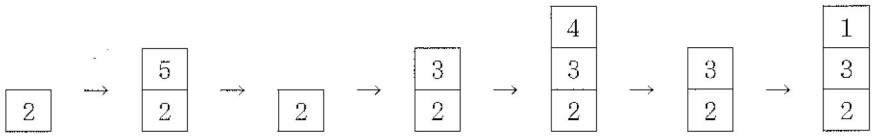
- データを入れ物の下から積みあげる
- 一番上にあるデータを取り出す

という２つの操作を行なうものである。なお，この入れ物のこともスタックと言う。

スタックにデータを出し入れする操作は，それぞれ「Push」，「Pop」と呼ばれる。

Push(n)	n というデータをスタックに入れる （「n を Push する」と言う）
Pop	スタックの一番上のデータを取り出す （「Pop する」と言う）

たとえば，Push(2)，Push(5)，Pop，Push(3)，Push(4)，Pop，Push(1)という手順で操作を行なうと，スタックの状態は次の図のように変化する。



学習指導要領（3）- 知・技 - ア
学習内容（3）- ア コンピュータの仕組みと処理

（問 1） Push(3)，Pop，Push(5)，Push(1)，Pop，Push(6)，Push(7)，Pop，Pop，Push(2)，Push(4)という手順で操作を行なうと，最終的にスタックの中のデータは下からどのように積みあがっているか。次の①～⑥の中から１つ選び，その番号を解答欄にマークしなさい。

①

4
2
1

②

2
7
3

③

2
6
5

④

2
7
1

⑤

4
2
5

⑥

4
2
3

学習指導要領（3）- 知・技 - イ
学習内容（3）- イ アルゴリズムとプログラム

（問 2） ある手順で操作を行なうと，最終的にスタックが次の状態になった。

4
6
3

このようになる手順として適切なものはA～Dのどれとどれか。下の①～⑥の中から１つ選び，その番号を解答欄にマークしなさい。

A Push(3), Push(1), Pop, Push(5), Push(6), Pop, Push(2), Pop, Push(4)

B Push(1), Pop, Push(3), Push(6), Push(5), Push(2), Pop, Pop, Push(4)

C Push(5), Pop, Push(3), Push(1), Pop, Push(6), Push(4), Push(2), Pop

D Push(3), Push(2), Pop, Push(5), Pop, Push(6), Push(1), Push(4), Pop

① AとB　② AとC　③ AとD

④ BとC　⑤ BとD　⑥ CとD

(問 3) 中置記法(一般的な式の記法)では，3種類のカッコ（ ），｛ ｝，［ ］が使われる。式中では，同じ種類の左右のカッコが対応していなければならない。たとえば，$\{(a+b)\times c-d\}\div e$ という式は，$(a+b\}$ の部分で左右のカッコの種類が異なっており，誤ったものである。このような，式中のカッコの対応の間違いを，スタックを使って調べることができる。その規則は次のとおりである。

- 最初，スタックはカラであるとする。
- 式を左から順番に処理していく。
- カッコ以外であれば，何もしない。
- 左カッコつまり［ ，｛ ，（ のいずれかならばPushする。
- 右カッコならばPopし，Popした左カッコの種類がこの右カッコの種類と異なるならば，誤りがあると判定する。

たとえば，$\{(a+b)\times c-d\}\div e$ という式の場合，｛ ，（ の順に左カッコがPushされる。これに続く a ，＋ ，b はカッコではないので何もしない。次に右カッコ ｝ があるので，Popすると，（ が得られる。この2つは種類が異なるので，カッコの対応が誤っていると判定する。

なお，通常の式では，内側から（ ），｛ ｝，［ ］の順番でカッコを使用するが，ここではこの順番は問わないことにする。これは，（3×{2＋1}）や（3×(2＋1)）なども正しいと判断することを意味する。また，$(a+b$ や $(a+b)\}$ のように，左右のカッコの数が異なる場合はここでは扱わないことにする。

上記の処理を実現するプログラムを，次の形式で作成することを考える。

- 変数とは，数値や文字のデータを記憶するための仕組みである。
- 変数にデータを記憶することを「代入する」と呼ぶ。変数にデータを代入するには，「変数名 ← 値」と記述する。
- プログラムで使用できる命令は次の表に示すもののみとする。

命令	意味
Push(a)	変数 a の値を Push する。
x ← Pop()	Pop し，Pop した値を変数 x に代入する。 カラのスタックを Pop した場合は，カラであることを表す文字列 "EMPTY" を変数 x に代入する。
a ← GetOpe()	式中で，まだ取り出されていない被演算子，演算子またはカッコのうち，最も左側のものを 1 つ取り出し，変数 a に代入する。 なお，取り出すものが残っていない場合は，"END" という文字列が変数 a に代入されるとする。
Print(z) Print("文字列")	変数 z の値もしくは「文字列」を画面に表示する。

条件分岐命令	意味

If 条件 Then
　処理
EndIf
	「条件」を満足したときのみ「処理」の部分を実行する。「処理」の部分は複数行であってもよい。
	なお，2 つの値が等しい条件は ＝ で，等しくない条件は ≠ で表すものとする。
	EndIf は If 命令の終端を表している。

繰り返し命令	意味
Repeat 　処理 While 条件	「処理」の部分を実行した後，「条件」が判定される。「処理」は，「条件」を満足している限り，繰り返し実行される。 「処理」の部分は複数行であってもよい。

［例 1］ 次のプログラムでは，式から演算子，被演算子またはカッコを 1 つ取り出し，それが 0 の場合はその値(0)を画面に表示する。0 以外の場合には，何も表示しない。

```
a ← GetOpe()
If  a = 0  Then
    Print(a)
EndIf
```

［例 2］ 次のプログラムでは，式から演算子，被演算子またはカッコが 1 つずつ取り出され，それが画面に表示される。つまり，式がそのまま表示される。

```
Repeat
    a← GetOpe()
    Print(a)
While  a ≠ "END"
```

これらの命令を用いて前述のカッコの対応を調べるプログラムを作成したところ，次のようになった。なお，プログラム中の＜…＞の部分には具体的なプログラムが書かれていないが，適切に書かれているものとする。このとき，空欄 イ ～ オ に当てはまる最も適切なものを，下の①～⑧の中から 1 つずつ選び，その番号を解答欄にマークしなさい。

```
Repeat
    a ← GetOpe()
    If  ＜a は左カッコ(, {, [ のいずれか＞  Then
        イ
    EndIf
    ウ
        エ
        オ
            Print("エラー")
        EndIf
    EndIf
    ＜ { } に対する処理＞
    ＜ [ ] に対する処理＞
While  a ≠ "END"
```

① If a = ")" Then
② If a = "(" Then
③ If b ≠ "(" Then
④ If b ≠ ")" Then
⑤ a ← Pop()
⑥ b ← Pop()
⑦ Push(a)
⑧ Push(b)

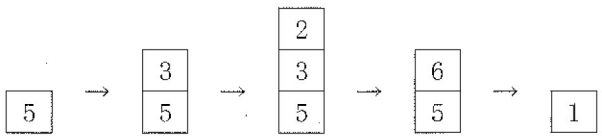
学習指導要領（3）- 知・技 - ウ
学習指導要領（3）- 思・判・表 - ウ
学習内容（3）- ウ モデル化とシミュレーション

（問 4） 大問〔Ⅱ〕の後置記法によって表わされた式は，被演算子がすべて数値である場合，スタックを利用して式の値を計算することができる。計算の規則は次の通りである。

- 最初，スタックはカラであるとする。
- 後置記法で書かれた式を左から順番に処理していく。
- 被演算子であれば Push する。
- 演算子であれば，スタックから 2 回 Pop する。そして，Pop した 2 つの数値に対して演算子に応じた計算をし，その結果を Push する。
- 式の最後まで計算し，最後にスタックに残った数値が求める値である。

後置記法で表わされた式 53－2× を上記の規則に基づいてスタックを利用して計算した場合，スタックの状態はどのように変化するか。次の①～⑥の中から 1 つ選び，その番号を解答欄にマークしなさい。

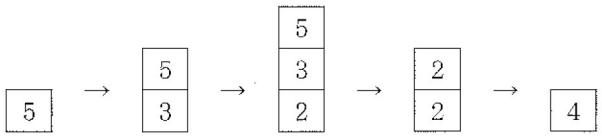
①



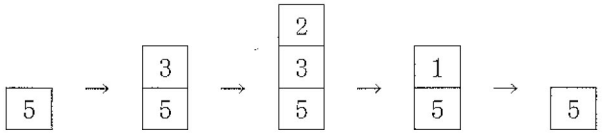
②

③

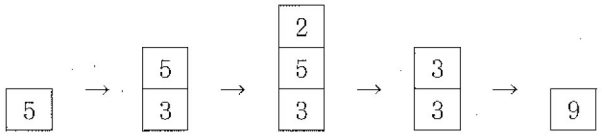
④



⑤



⑥



学習指導要領（3）- 知・技 - イ
学習指導要領（3）- 思・判・表 - イ
学習内容（3）- イ アルゴリズムとプログラム

（問 5）（問 4）で説明した手順を実現するプログラムを，（問 3）と同じ形式で作成したところ，次のようになった。なお，式の終わりを表わす"END"という文字列は，被演算子でも演算子でもないと扱われるものとする。また，プログラム中の＜…＞の部分には具体的なプログラムが書かれていないが，適切に書かれているものとする。このとき，空欄 カ ～ コ に当てはまる最も適切なものを，次の①～⑧の中から１つずつ選び，その番号を解答欄にマークしなさい。

```
Repeat
    a ← GetOpe()
    If  ＜a は被演算子＞  Then
        カ
    EndIf
    If  ＜a は演算子＞  Then
        キ
        ク
        If  a = "+"  Then
            z ← x + y
        EndIf
        If  a = "−"  Then
            z ← x − y
        EndIf
        ＜a = "×" の場合の処理＞
        ＜a = "÷" の場合の処理＞
        ケ
    EndIf
While  a ≠ "END"
コ
Print(z)
```

① a ← Pop()　② x ← Pop()
③ y ← Pop()　④ z ← Pop()
⑤ Push(a)　⑥ Push(x)
⑦ Push(y)　⑧ Push(z)