問2　構文解析に関する次の記述を読んで，設問1～4に答えよ。

宣言部と実行部からなる図 1 のような記述をするプログラム言語がある。その構文規則を，括弧記号で表記を拡張したBNFによって，図2のように定義した。

```
short  aa  ;          ┐
long   b1  ;          ├ 宣言部
long   c   ;          ┘
aa  =  3  ;           ┐
b1  =  aa  -  1  ;    ├ 実行部
c   =  aa  +  2  *  b1  ;  ┘
```

図1　プログラムの記述例

図 2 において，引用符「‘」と「’」で囲まれた記号や文字列，＜数＞，及び＜識別子＞は終端記号を表す。そのほかの「＜」と「＞」で囲まれた名前は非終端記号を表す。＜数＞は 1 文字以上の数字の列を表し，＜識別子＞は英字で始まる 1 文字以上の英字又は数字からなる文字列を表す。

また，A｜B は A と B のいずれかを選択することを表し，{A} は A を 0 回以上繰り返すことを表す。

＜プログラム＞ ：：＝ ＜宣言部＞ ＜実行部＞
＜宣言部＞ ：：＝ ＜宣言部記述＞ {＜ [ア] ＞}
＜実行部＞ ：：＝ ＜文＞ {＜文＞}
＜宣言部記述＞ ：：＝ ＜宣言記述子＞ ＜ [イ] ＞ ‘;’
＜宣言記述子＞ ：：＝ ‘short’ ｜ ‘long’
＜文＞ ：：＝ ＜識別子＞ ‘=’ ＜式＞ ‘;’
＜式＞ ：：＝ ＜項＞ {‘+’ ＜項＞｜ ‘-’ ＜項＞}
＜項＞ ：：＝ ＜ [ウ] ＞ {‘*’ ＜因子＞｜ ‘/’ ＜因子＞}
＜因子＞ ：：＝ ＜数＞｜＜識別子＞

図2　構文規則

例えば，図1の最初の行“short　aa　;”は，図2の＜宣言部記述＞の定義に従っていて，＜宣言記述子＞と‘short’，＜ [イ] ＞と‘aa’，更に‘;’同士がそれぞれ対応していることが分かる。

〔字句解析関数の定義〕

プログラム記述が図 2 の構文規則に従っているかどうかを検査するプログラムを作成するために，字句を先頭から順番に抽出し，その種類を判定する関数 gettoken() を定義する。ここでいう字句とは，構文規則における終端記号である。字句と字句は，空白や改行文字で区切られている。空白や改行文字は，字句そのものには含まれない。字句の種類と関数 gettoken()の戻り値の対応を表に示す。

なお，‘short’と‘long’は＜識別子＞には含まれない。また，いずれの終端記号にも該当しない字句やプログラム記述の終わりを検出した場合の戻り値も定義する。

表　字句の種類と関数 gettoken( )の戻り値の対応

字句の種類	戻り値
‘short’	‘S’
‘long’	‘L’
＜数＞	‘N’
＜識別子＞	‘I’
‘=’，‘+’，‘-’，‘*’，‘/’及び‘;’	左の各字句に同じ
いずれにも該当しない字句	‘?’
プログラム記述の終わり	‘$’

〔構文解析プログラム〕

図 3 は，図 2 の構文規則に従って，＜文＞及び＜式＞の構文を検査するプログラムである。プログラムの前提条件を次に示す。

(1)　＜文＞，＜式＞及び＜項＞の構文解析を行う関数をそれぞれ bun(), shiki()及び kou()とする。これらの関数の戻り値は，構文が正しい場合は 0，エラーが検知された場合は−1 である。

(2)　構文解析を行う各関数実行開始時の変数 token の値は，検査の対象となる文字列の最初の字句に対する関数 gettoken()の戻り値である。

```
function bun()
  if ( token と'I'が等しい ) then
    token ← gettoken()
  else
    return -1
  endif
  if ( [ エ ] ) then
    token ← gettoken()
  else
    return -1
  endif
  if ( shiki() と -1 が等しい ) then return -1 endif
  if ( token と';'が等しい ) then
    token ← gettoken()
  else
    return -1
  endif
  return 0
endfunction

function shiki()
  if ( [ オ ] ) then return -1 endif
  while ( token と'+'が等しい 又は token と'-'が等しい )
    if ( token と'+'が等しい ) then
      token ← gettoken()
      if ( kou()と -1 が等しい ) then return -1 endif
    elseif ( token と'-'が等しい ) then
      token ← gettoken()
      if ( kou()と -1 が等しい ) then return -1 endif
    else
      return -1
    endif
  endwhile
  return 0
endfunction
```

図３　＜文＞と＜式＞の構文解析関数

設問1　図2中の［　ア　］～［　ウ　］に入れる適切な非終端記号又は終端記号の名前を答えよ。

設問2　次のプログラム記述には，図2で示した構文規則に反するエラーが幾つか含まれている。構文規則に反するエラーを含む行の番号をすべて答えよ。

```
short  abc  ;                  ・・・・・ ①
short  def  ghi  ;             ・・・・・ ②
long  mno  ;                   ・・・・・ ③
abc  =  def  +  34  ;          ・・・・・ ④
ghi  =  -  2  *  mno  ;        ・・・・・ ⑤
mno  =  abc  /  0  ;           ・・・・・ ⑥
xyz  =  def  -  7  ;           ・・・・・ ⑦
```

設問3　図3中の［　エ　］，［　オ　］に入れる適切な字句を答えよ。

設問4　“d = a * (3 + b) ;”のように，式の演算子の評価順序を明示的に記述するため，「(」及び「)」を使えるように構文規則を拡張したい。これについて，(1)，(2)に答えよ。

(1)　図2の構文規則の中の＜因子＞の行を，次のように書き換えた。

［　カ　］に入れる適切な字句を答えよ。

＜因子＞ ：：＝ ＜数＞｜＜識別子＞｜ ‘(’ ［　カ　］ ‘)’

(2)　この構文規則の拡張によって，関数 gettoken()も修正する必要がある。どのような修正が必要か，35字以内で述べよ。