次の問８は必須問題です。必ず解答してください。

問８　次のプログラムの説明及びプログラムを読んで，設問１～３に答えよ。

事務計算においては，数値を見やすく表示（印字）するために，例えば３桁ごとに区切りの“,”を挿入するなどの編集処理がよく行われる。

関数 Edit は，指定された編集パターンに従って，数値を編集するプログラムである。表１に，関数 Edit を用いた編集例を示す。例１では，３桁ごとに区切りの“,”を挿入している。例２では，例１の編集に加え，上位の空いた桁を“*”で埋めている。例３では，数値の右端から２桁目と３桁目の間に“.”を挿入している。

表１　関数 Edit を用いた編集例

		編集パターン		
		例１	例２	例３
		“␣□□,□□□”	“*□□,□□□”	“␣□□■.□□”
数値	123	“␣␣␣␣123”	“****123”	“␣␣␣1.23”
	1234	“␣␣1,234”	“**1,234”	“␣␣12.34”
	12345	“␣12,345”	“*12,345”	“␣123.45”

ここで，編集パターン中の文字“□”及び“■”は，数字と対応付けされた制御文字を表している。また，“␣”は空白文字を表している。

h28au08データ構造及びアルゴリズム_数値の編集.pdf

〔プログラムの説明〕

(1) 関数 Edit は，次の形式で呼び出され，二つの引数をもつ。

関数: Edit(文字型: Pattern[], 文字型: Value[])

Pattern[] には，編集パターンの文字列が格納されている。Value[] には，編集する数値を表す文字列が格納されている。各配列の添字は，0 から始まる。文字列 Pattern[] の i 番目の文字は Pattern[i − 1] と表記する。文字列 Value[] についても同様である。

(2) Pattern[] は，1 文字以上から成る文字列であって，表示可能な図形文字及び制御文字（“□” 及び “■”）から構成される。

(3) Value[] は，数値を表す文字列であって，数字 “0” ～ “9” の並びの後に，数値が正又は 0 なら “+” を，負なら “-” を付加した形式である。数字の個数は，Pattern[] 中の文字 “□” 及び “■” の個数と一致するように，必要であれば前方に “0” を付加する。例えば，Pattern[] の内容が “*□□,□■□” のとき，Value[] には，数値が 123 なら “00123+”，0 なら “00000+”，−123 なら “00123-” を指定する。

(4) 関数 Edit は，Value[] で与えられた数値を Pattern[] に従って編集し，編集結果で Pattern[] を置き換える。

〔編集方法〕

Pattern[] 中の各文字について，先頭から順に 1 文字ずつ，次の ① ～ ③ のいずれか一つの操作を実行していく。

① 関数 Edit が呼び出されたときの Pattern[] 中の先頭の文字（以下，fill 文字という）で置き換える。

② Value[] 中の対応する桁の数字で置き換える。

③ 置き換えないで，そのまま残す。

(5) 論理型変数 signif は，on 又は off の値を取る。この変数の実行開始時の値は off であり，Value[] 中に最上位から “0” が連続した後に “0” でない数字が見つかると on になる，などの使い方をする。

(6) 関数 Edit が呼び出されるとき，各引数には正しい値が設定されているものとする。

〔プログラム〕

```
○関数: Edit(文字型: Pattern[], 文字型: Value[])
○文字型: fill
○論理型: signif
○整数型: p, v

・fill ← Pattern[0]
・signif ← off
・v ← 0
■ p: 0, p < Length(Pattern[]), 1   /* Length()は引数の文字列長を返す */
│▲ Pattern[p] = "□"  or Pattern[p] = "■"  /* 表2のケース1～7の処理 */
││ ┌ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ┐
││ ¦ 現在の変数・配列要素の内容が，表2のケース1～7の            ¦
││ ¦ どれに該当するかを決定し，そのケースに従って                ¦
││ ¦ Pattern[p] と signif の更新処理を行う。                   ¦
││ └ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ┘
││ ・v ← v + 1
│├──                                   /* 表2のケース8，9の処理 */
││▲ signif = off
│││ ・Pattern[p] ← fill
││▼
│▼
■
```

表2　現在の変数・配列要素の内容に応じた更新処理

	現在の変数・配列要素の内容				更新処理	
ケース	Pattern[p]	signif	Value[v]	Value[v+1]	Pattern[p]	signif
1	"□"	off	"0"		fill 文字	off
2	"■"	off	"0"	"+" 以外	fill 文字	on
3	"■"	off	"0"	"+"	fill 文字	off
4	"□" 又は "■"	off	"1" ～ "9"	"+" 以外	Value[v]	on
5	"□" 又は "■"	off	"1" ～ "9"	"+"	Value[v]	off
6	"□" 又は "■"	on	"0" ～ "9"	"+" 以外	Value[v]	on
7	"□" 又は "■"	on	"0" ～ "9"	"+"	Value[v]	off
8	"□" と "■" 以外	off			fill 文字	off
9	"□" と "■" 以外	on			そのまま残す	on

注記　網掛け部分は，内容を判定しない。

設問1　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

引数 Pattern[] 及び Value[] に幾つかのデータを与えて，関数 Edit を実行した結果を，表3に示す。

表3　関数 Edit の実行結果

実行前の内容		実行後の内容
Pattern[]	Value[]	Pattern[]
“␣␣□□,□□□”	“01234+”	“␣␣␣1,234”
“*□□,□□□#”	“00000+”	［　a　］
“*□□□.□□#”	“00012-”	［　b　］
“*□□■.□□#”	“00012+”	［　c　］

aに関する解答群

ア　“*******#”　　イ　“********”

ウ　“******0#”　　エ　“******0*”

b，cに関する解答群

ア　“*****12#”　　イ　“*****12*”

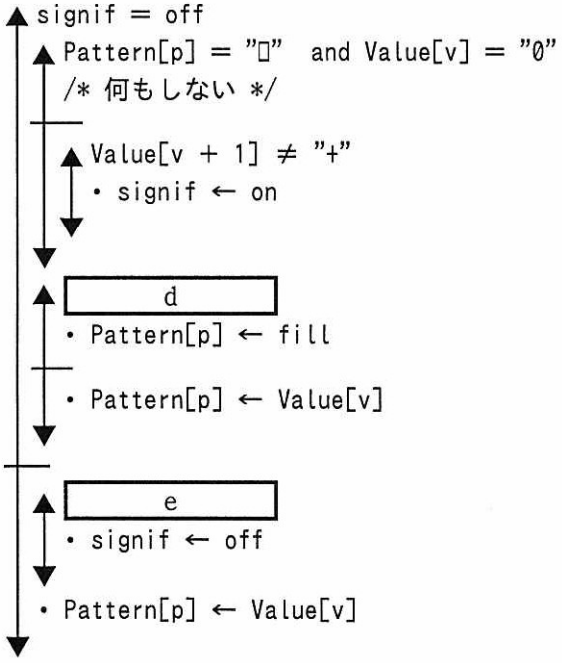
ウ　“****.12#”　　エ　“****.12*”

オ　“***0.12#”　　カ　“***0.12*”

設問２　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

プログラム中の破線で囲んだ部分の処理（表２のケース１～７の処理）を，詳細なプログラムとして記述すると，次のようになる。

〔プログラム中の［　　］部分の処理〕



d，eに関する解答群

ア　"1" ≦ Value[v] and Value[v] ≦ "9"　　イ　Value[v] = "0"

ウ　Value[v + 1] = "-"　　エ　Value[v + 1] = "+"

オ　Value[v + 1] ≠ "-"　　カ　Value[v + 1] ≠ "+"

設問３　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

関数 Edit では，例えば，fill 文字を“␣”とする編集パターンを指定することによって，数値が正なら“␣␣1,234␣”，負なら“␣␣1,234-”と編集することができる。表２のケース１～７のうち，数値が正なら数値の後に続く文字を fill 文字で置き換えるために用意されたケースは［　f　］である。

fに関する解答群

ア　2，4及び7　　イ　3，5及び7

ウ　4及び7　　エ　5及び7