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。

〔プログラムの説明〕

関数 BitapMatch は，Bitap 法を使って文字列検索を行うプログラムである。

Bitap 法は，検索対象の文字列（以下，対象文字列という）と検索文字列の照合に，個別の文字ごとに定義されるビット列を用いるという特徴をもつ。

なお，本問では，例えば 2 進数の 16 ビット論理型の定数 0000000000010101 は，上位の 0 を省略して “10101”B と表記する。

(1) 関数 BitapMatch は，対象文字列を Text[] に，検索文字列を Pat[] に格納して呼び出す。配列の要素番号は１から始まり，Text[] の i 番目の文字は Text[i] と表記する。Pat[] についても同様に i 番目の文字は Pat[i] と表記する。対象文字列と検索文字列は，英大文字で構成され，いずれも最長 16 文字とする。

対象文字列 Text[] が “AACBBAACABABAB”，検索文字列 Pat[] が “ACABAB” の場合の格納例を，図１に示す。

要素番号	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Text[]	A	A	C	B	B	A	A	C	A	B	A	B	A	B

要素番号	1	2	3	4	5	6
Pat[]	A	C	A	B	A	B

図１　対象文字列と検索文字列の格納例

(2) 関数 BitapMatch は，関数 GenerateBitMask を呼び出す。

関数 GenerateBitMask は，文字 “A” ～ “Z” の文字ごとに，検索文字列に応じたビット列（以下，ビットマスクという）を生成し，要素数 26 の 16 ビット論理型配列 Mask[] に格納する。Mask[1] には文字 “A” に対するビットマスクを，Mask[2]

には文字“B”に対するビットマスクを格納する。このように Mask[1]～ Mask[26] に文字“A”～“Z”に対応するビットマスクを格納する。

関数 GenerateBitMask は，Mask[] の全ての要素を“0”B に初期化した後，1以上で Pat[] の文字数以下の全ての i に対して，Pat[i] の文字に対応する Mask[] の要素である Mask[Index(Pat[i])] に格納されている値の，下位から数えて i 番目のビットの値を1にする。

関数 Index は，引数にアルファベット順で n 番目の英大文字を設定して呼び出すと，整数 n（1 ≦ n ≦ 26）を返す。

(3) 図 1 で示した，Pat[] が“ACABAB”の例の場合，関数 GenerateBitMask を実行すると，Mask[] は図2のとおりになる。

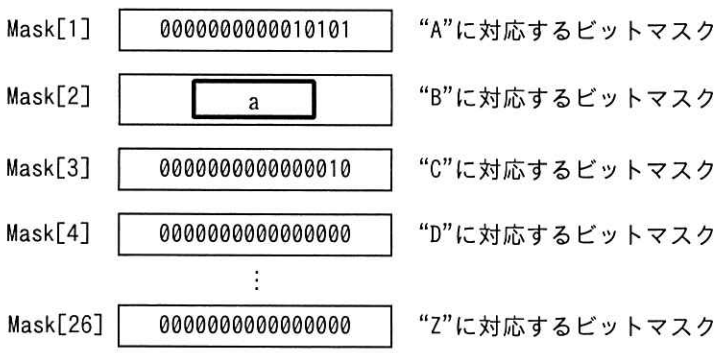

図2　図1で示した Pat[] に対する Mask[] の値

(4) 関数 GenerateBitMask の引数と返却値の仕様は，表1のとおりである。

表1　関数 GenerateBitMask の引数と返却値の仕様

引数／返却値	データ型	入力／出力	説明
Pat[]	文字型	入力	検索文字列が格納されている1次元配列
Mask[]	16ビット論理型	出力	文字“A”～“Z”に対応するビットマスクが格納される1次元配列
返却値	整数型	出力	検索文字列の文字数

〔プログラム 1〕

```
○整数型関数: GenerateBitMask(文字型: Pat[], 16 ビット論理型: Mask[])
○整数型: i, PatLen

・ PatLen ← Pat[] の文字数
■ i: 1, i ≦ 26, 1
|  ・ Mask[i] ← [    b    ]        /* 初期化 */
■
■ i: 1, i ≦ PatLen, 1
|  ・ Mask[Index(Pat[i])] ← [    c    ] と
|                    Mask[Index(Pat[i])] とのビットごとの論理和
■
・ return (PatLen)
```

設問 1　プログラムの説明及びプログラム 1 中の [　　] に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　0000000000000101　　イ　0000000000101000

ウ　0001010000000000　　エ　1010000000000000

b に関する解答群

ア　“0”B

イ　“1”B

ウ　“1”B を PatLen ビットだけ論理左シフトした値

エ　“1”B を (PatLen − 1) ビットだけ論理左シフトした値

オ　“1111111111111111”B

c に関する解答群

ア　“1”B を (i − 1) ビットだけ論理左シフトした値

イ　“1”B を i ビットだけ論理左シフトした値

ウ　“1”B を (PatLen − 1) ビットだけ論理左シフトした値

エ　“1”B を PatLen ビットだけ論理左シフトした値

オ　“1”B

〔関数 BitapMatch の説明〕

(1) Text[] と Pat[] を受け取り，Text[] の要素番号の小さい方から Pat[] と一致する文字列を検索し，見つかった場合は，一致した文字列の先頭の文字に対応する Text[] の要素の要素番号を返し，見つからなかった場合は，−1 を返す。

(2) 図 1 の例では，Text[7] ～ Text[12] の文字列が Pat[] と一致するので，7 を返す。

(3) 関数 BitapMatch の引数と返却値の仕様は，表 2 のとおりである。

表 2　関数 BitapMatch の引数と返却値の仕様

引数／返却値	データ型	入力／出力	説明
Text[]	文字型	入力	対象文字列が格納されている 1 次元配列
Pat[]	文字型	入力	検索文字列が格納されている 1 次元配列
返却値	整数型	出力	対象文字列中に検索文字列が見つかった場合は，一致した文字列の先頭の文字に対応する対象文字列の要素の要素番号を，検索文字列が見つからなかった場合は，−1 を返す。

〔プログラム 2〕

```
○整数型関数: BitapMatch(文字型: Text[], 文字型: Pat[])
○16 ビット論理型: Goal, Status, Mask[26]
○整数型: i, TextLen, PatLen
・ TextLen ← Text[] の文字数
・ PatLen ← GenerateBitMask(Pat[], Mask[])
・ Status ← "0"B
・ Goal ← "1"B を (PatLen − 1) ビットだけ論理左シフトした値
■ i: 1, i ≦ TextLen, 1
  ・ Status ← Status を 1ビットだけ論理左シフトした値と          ← α
                                        "1"B とのビットごとの論理和
  ・ Status ← Status と Mask[Index(Text[i])] とのビットごとの論理積  ← β
  ▲ Status と Goal とのビットごとの論理積 ≠ "0"B
  ▼   ・ return (i − PatLen + 1)
■
・ return (−1)
```

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。

図1で示したとおりに，Text[] と Pat[] に値を格納し，関数 BitapMatch を実行した。プログラム2の行βを実行した直後の変数 i と配列要素 Mask[Index(Text[i])] と変数 Status の値の遷移は，表3のとおりである。

例えば，i が1のときに行βを実行した直後の Status の値は“1”B であることから，i が2のときに行αを実行した直後の Status の値は，“1”B を1ビットだけ論理左シフトした“10”B と“1”B とのビットごとの論理和を取った“11”B となる。次に，i が2のときに行βを実行した直後の Status の値は，Mask[Index(Text[2])] の値が“10101”B であることを考慮すると，［ d ］となる。

同様に，i が8のときに行βを実行した直後の Status の値が“10”B であるということに留意すると，i が9のときに行αを実行した直後の行βで参照する Mask[Index(Text[9])] の値は［ e ］であるので，行βを実行した直後の Status の値は［ f ］となる。

表3　図1の格納例に対してプログラム2の行βを実行した直後の配列要素 Mask[Index(Text[i])] と変数 Status の値の遷移

i	1	2	…	8	9	…
Mask[Index(Text[i])]	“10101”B	“10101”B	…	“10”B	e	…
Status	“1”B	d	…	“10”B	f	…

d～f に関する解答群

ア　“0”B　　イ　“1”B　　ウ　“10”B　　エ　“11”B

オ　“100”B　　カ　“101”B　　キ　“10101”B

設問３　関数 GenerateBitMask の拡張に関する，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。ここで，プログラム 3 中の［　b　］には，設問１の［　b　］の正しい答えが入っているものとする。

表 4 に示すような正規表現を検索文字列に指定できるように，関数 GenerateBitMask を拡張し，関数 GenerateBitMaskRegex を作成した。

表４　正規表現

記号	説明
[]	[] 内に記載されている文字のいずれか 1 文字に一致する文字を表す。例えば，“A[XYZ]B”は，“AXB”，“AYB”，“AZB”を表現している。

〔プログラム 3〕

```
○整数型関数: GenerateBitMaskRegex(文字型: Pat[],
                                   16 ビット論理型: Mask[])
○整数型: i, OriginalPatLen, PatLen, Mode

・ OriginalPatLen ← Pat[] の文字数
・ PatLen ← 0
・ Mode ← 0
■ i: 1, i ≦ 26, 1
  ・ Mask[i] ← [   b   ]        /* 初期化 */
■
■ i: 1, i ≦ OriginalPatLen, 1
  ▲ Pat[i] = "["
    ・ Mode ← 1
    ・ PatLen ← PatLen + 1
  ――――
    ▲ Pat[i] = "]"
      ・ Mode ← 0
    ――――
      ▲ Mode = 0
        ・ PatLen ← PatLen + 1

      ・ Mask[Index(Pat[i])] ← "1"B を (PatLen − 1) ビットだけ
           論理左シフトした値と Mask[Index(Pat[i])] とのビットごとの論理和
■
・ return (PatLen)
```

Pat[]に“AC[BA]A[ABC]A”を格納して，関数 GenerateBitMaskRegex を呼び出した場合を考える。この場合，文字“A”に対応するビットマスクである Mask[1]は［　g　］となり，関数 GenerateBitMaskRegex の返却値は［　h　］となる。また，Pat[] に格納する文字列中において [] を入れ子にすることはできないが，誤って Pat[] に“AC[B[AB]AC]A”を格納して関数 GenerateBitMaskRegex を呼び出した場合，Mask[1] は［　i　］となる。

g，iに関する解答群

ア　“1001101”B　　イ　“1010100001”B

ウ　“1011001”B　　エ　“101111”B

オ　“110011”B　　カ　“111101”B

hに関する解答群

ア　4　　イ　6　　ウ　9　　エ　13