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 BMMatch は，Boyer-Moore-Horspool 法（以下，BM 法という）を用いて，文字列検索を行うプログラムである。BM 法は，検索文字列の末尾の文字から先頭に向かって，検索対象の文字列（以下，対象文字列）と 1 文字ずつ順に比較していくことで照合を行う。比較した文字が一致せず，照合が失敗した際には，検索文字列中の文字の情報を利用して，次に照合を開始する対象文字列の位置を決定する。このようにして明らかに不一致となる照合を省き，高速に検索できる特徴がある。

(1) 対象文字列を Text[]，検索文字列を Pat[] とする。ここで，配列の添字は 1 から始まり，文字列 Text[] の i 番目の文字は Text[i] と表記される。Pat[] についても同様に i 番目の文字は Pat[i] と表記される。また，対象文字列と検索文字列は，英大文字から構成される。

例えば，対象文字列 Text[] が“ACBBMACABABC”，検索文字列 Pat[] が“ACAB”の場合の例を図１に示す。

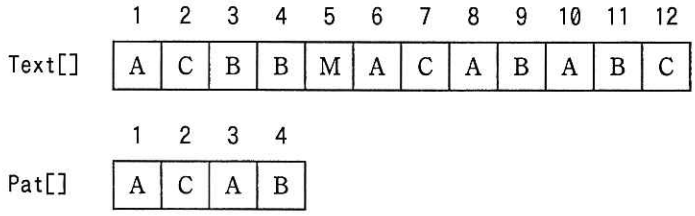

図１　対象文字列と検索文字列の格納例

(2) 関数 BMMatch では，照合が失敗すると，次に照合を開始する位置まで検索文字列を移動するが，その移動量を格納した要素数 26 の配列 Skip[] をあらかじめ作成しておく。Skip[1] に文字“A”に対応する移動量を，Skip[2] に文字“B”に対応する移動量を格納する。このように，Skip[1] ～ Skip[26] に文字“A”～“Z”

に対応する移動量を格納する。ここで，検索文字列の長さを PatLen とすると，移動量は次のようになる。

① 検索文字列の末尾の文字 Pat[PatLen] にだけ現れる文字と，検索文字列に現れない文字に対応する移動量は，PatLen である。

② 検索文字列の Pat[1] から Pat[PatLen － 1] に現れる文字に対応する移動量は，その文字が，検索文字列の末尾から何文字目に現れるかを数えた文字数から 1 を引いた値とする。ただし，複数回現れる場合は，最も末尾に近い文字に対応する移動量とする。

(3) 図 1 で示した Pat[] の例の場合，次の①～④に示すように，Skip[] は図 2 のとおりになる。

① 文字“A”は検索文字列の末尾から 2 文字目（Pat[3]）と 4 文字目（Pat[1]）に現れるので，末尾に近い Pat[3]に対応する移動量の 1（＝2－1）となる。

② 文字“B”は検索文字列の末尾の文字にだけ現れるので，移動量は PatLen（＝4）となる。

③ 文字“C”は検索文字列の末尾から 3 文字目（Pat[2]）に現れるので，移動量は 2（＝3－1）となる。

④ “A”，“B”及び“C”以外の文字については検索文字列に現れないので，移動量は PatLen（＝4）となる。

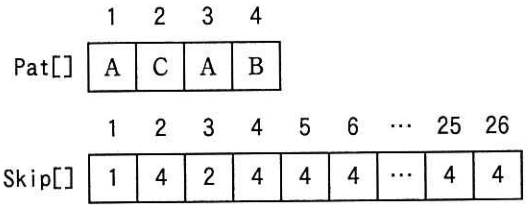

図 2　図 1 の格納例における Skip[] の値

(4) 図1の例で照合する場合の手順は，次の①～⑨となり，その流れを図3に示す。
この例では，PatLen＝4なので，検索文字列の末尾の文字はPat[4]である。

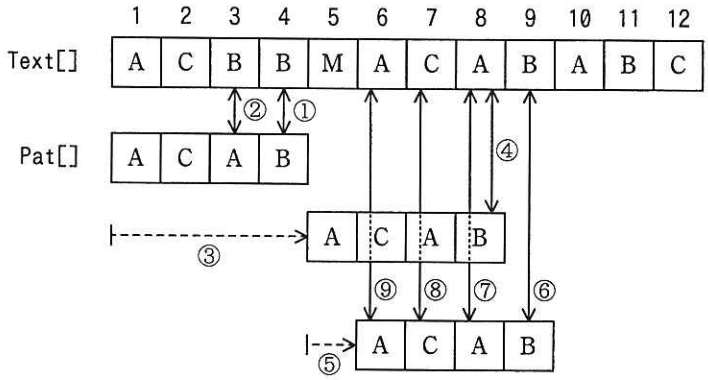

図3　図1の場合の照合手順

① Text[4]とPat[4]を比較する。Text[4]とPat[4]は同じ文字“B”である。

② Text[3]とPat[3]を比較する。Text[3]の“B”とPat[3]の“A”は異なる文字である。

③ ①で検索文字列の末尾の文字Pat[4]と比較したText[4]を基準に，Text[4]の文字“B”に対応する移動量であるSkip[2]の値4だけPat[]を右側に移動し，Text[8]とPat[4]の比較に移る。

④ Text[8]とPat[4]を比較する。Text[8]の“A”とPat[4]の“B”は異なる文字である。

⑤ ④で検索文字列の末尾の文字Pat[4]と比較したText[8]を基準に，Text[8]の文字“A”に対応する移動量であるSkip[1]の値1だけPat[]を右側に移動し，Text[9]とPat[4]の比較に移る。

⑥ Text[9]とPat[4]を比較する。Text[9]とPat[4]は同じ文字“B”である。

⑦ Text[8]とPat[3]を比較する。Text[8]とPat[3]は同じ文字“A”である。

⑧ Text[7]とPat[2]を比較する。Text[7]とPat[2]は同じ文字“C”である。

⑨ Text[6]とPat[1]を比較する。Text[6]とPat[1]は同じ文字“A”である。

⑥～⑨の比較で，対象文字列 Text[]の連続した一部分が検索文字列 Pat[]に完全に一致したので，検索は終了する。

〔関数 BMMatch の引数と返却値〕

関数 BMMatch の引数と返却値の仕様は，次のとおりである。

引数名／返却値	データ型	意味
Text[]	文字型	対象文字列が格納されている１次元配列
TextLen	整数型	対象文字列の長さ（１以上）
Pat[]	文字型	検索文字列が格納されている１次元配列
PatLen	整数型	検索文字列の長さ（１以上）
返却値	整数型	対象文字列中に検索文字列が見つかった場合は，１以上の値を返す。 検索文字列が見つからなかった場合は，−1 を返す。

関数 BMMatch では，次の関数 Index を使用する。

〔関数 Index の仕様〕

引数にアルファベット順で n 番目の英大文字を与えると，整数 n(1≦n≦26)を返却値とする。

〔プログラム〕

```
○整数型関数: BMMatch(文字型: Text[], 整数型: TextLen,
                      文字型: Pat[], 整数型: PatLen)
○整数型: Skip[26], PText, PPat, PLast, I

■ I: 1, I ≦ 26, 1
  • Skip[I] ← [ a ]
■
■ I: 1, I ≦ PatLen − 1, 1 ←――――――――――― γ
  • Skip[Index(Pat[I])] ← [ b ]
■
• PLast ← PatLen
■ PLast ≦ TextLen
  • PText ← PLast ←――――――――――――――― α
  • PPat ← PatLen
  ■ Text[PText] = Pat[PPat]
    ▲ PPat = 1 ←――――――――――――――― β
    ▼ • return (PText)
    • PText ← PText − 1
    • PPat ← PPat − 1
  ■
  • PLast ← PLast + Skip[Index(Text[PLast])]
■
• return (−1)
```

E 後

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　0　　イ　1　　ウ　I － PatLen

エ　PatLen　　オ　PatLen － 1　　カ　PatLen － I

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4のように，Text[]に“ABCXBBACABACADEC”，TextLenに16，Pat[]に“ABAC”，PatLenに4を格納し，BMMatch (Text[], TextLen, Pat[], PatLen)を呼び出した。プログラムが終了するまでにαは［　c　］回実行され，βは［　d　］回実行される。またこの場合，関数BMMatchの返却値は［　e　］である。

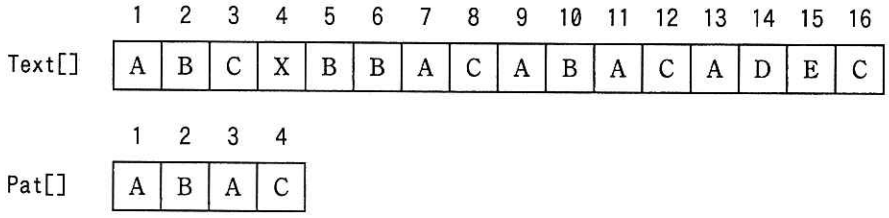

図4　対象文字列と検索文字列

c～eに関する解答群

ア　3　　イ　4　　ウ　5　　エ　6　　オ　7

カ　8　　キ　9　　ク　10　　ケ　11　　コ　12

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。ここで，プログラム中の［ a ］と［ b ］には正しい答えが入っているものとする。

関数 BMMatch 中のγの処理を

I: PatLen − 1, I ≧ 1, −1

に変更した場合，関数 BMMatch は［ f ］。

fに関する解答群

ア　対象文字列中に，検索文字列が含まれていないのに，1 以上の値を返す場合がある

イ　対象文字列中に，検索文字列が含まれているのに，−1 を返す場合がある

ウ　正しい値を返す