問2　文字列を圧縮するアルゴリズムに関する次の記述を読んで，設問1～4に答えよ。

データを圧縮するアルゴリズムの一つにランレングス法がある。ランレングス法とは，同じデータが連続して現れる箇所を，そのデータと連続している回数との組に変換する方法である。文字“a”～“z”だけから成る文字列を圧縮する方法として，圧縮の表現形式の違う二つのプログラムを比較検討する。

圧縮前と圧縮後のデータを管理する方法として配列を用いる。配列の各要素には，文字データの場合は8ビット表現の文字コードが，数値データの場合は0～255の整数が格納される。圧縮前の配列をin，圧縮後の配列をoutとする。配列inの大きさは文字列の長さと等しく，2以上，255以下である。配列outには圧縮後のデータを格納する十分な領域が確保されている。配列の添字は0から始まる。

〔圧縮方法その1〕

圧縮の表現形式として，[圧縮対象文字][連続回数]を用いる方法の処理手順を次の(1)～(5)に，そのプログラムを図1に示す。例えば，文字列“abbcccddddeeeee”を圧縮すると，a1b2c3d4e5となる。ここで，a～zは文字データを表し，数字は対応する数値データを表す。

(1)　配列inの初めの1文字を変数prevに取り出す。連続回数を1にする。

(2)　配列inから次の1文字を取り出し，変数prevと比較する。配列inから取り出す文字がない場合，処理手順(5)へ進む。

(3)　比較した二つの文字が等しい場合，連続回数に1を加える。等しくない場合，変数prevと連続回数を配列outに追加し，(2)で取り出した文字を変数prevにコピーして，連続回数を1に戻す。

(4)　処理手順(2)に戻る。

(5)　変数prevと連続回数を配列outに追加する。

図1中の関数encode1はプログラムのメイン処理であり，配列outに追加されたデータの大きさを返す。

```
function encode1( in, out )
  prev ← in[0]
  runLength ← 1
  k ← 0
  for ( i を 1 から(配列 in の大きさ−1)まで 1 ずつ増やす )
    if ( [ ア ] )
      runLength ← runLength + 1
    else
      out[k] ← prev
      out[k+1] ← runLength
      k ← [ イ ]
      prev ← in[i]
      runLength ← 1
    endif
  endfor
  out[k] ← prev
  out[k+1] ← runLength
  k ← [ イ ]
  return k
endfunction
```

図 1　圧縮方法その 1 のプログラム

〔圧縮方法その 2〕

圧縮の表現形式として，同じ文字が 4 回以上連続する場合に[圧縮対象文字][圧縮表現文字][連続回数]を用い，3 回以下の場合はそのままとする方法の処理手順を次の(1)～(5)に，そのプログラムを図 2 に示す。圧縮表現文字には，圧縮対象文字と区別するために，圧縮対象文字として使用されない文字を使う。ここでは“*”を圧縮表現文字とする。例えば，文字列“abbcccddddeeeee”を圧縮すると，abbcccd*4e*5 となる。

(1)　配列 in の初めの 1 文字を変数 prev に取り出す。連続回数を 1 にする。

(2)　配列 in から次の 1 文字を取り出し，変数 prev と比較する。配列 in から取り出す文字がない場合，処理手順(5)へ進む。

(3)　比較した二つの文字が等しくない場合，変数 prev の連続回数だけの繰返しを表す圧縮表現を配列 out に追加し，(2)で取り出した文字を変数 prev にコピーして，連続回数を 1 に戻す。等しい場合，連続回数に 1 を加える。

(4)　処理手順(2)に戻る。

(5)　変数 prev の連続回数だけの繰返しを表す圧縮表現を配列 out に追加する。

図 2 中の関数 encode2 はプログラムのメイン処理であり，配列 out に追加されたデータの大きさを返す。関数 outputRun は，prev が runLength 回繰り返すことを表す圧縮表現を配列 out の添字 k の位置に追加し，次の追加位置を返す。

```
function encode2( in, out )
  prev ← in[0]
  runLength ← 1
  k ← 0
  for ( i を 1 から(配列 in の大きさ−1)まで 1 ずつ増やす )
    if ( [ ウ ] )
      k ← outputRun( prev, runLength, k, out )
      [ エ ]
      runLength ← 1
    else
      runLength ← runLength + 1
    endif
  endfor
  k ← outputRun( prev, runLength, k, out )
  return k
endfunction

function outputRun( prev, runLength, k, out )
  if ( [ オ ] )
    for ( i を 1 から runLength まで 1 ずつ増やす )
      out[k] ← prev
      k ← k + 1
    endfor
  else
    out[k] ← prev
    out[k+1] ← "*"
    out[k+2] ← runLength
    k ← k + 3
  endif
  return k
endfunction
```

図 2　圧縮方法その 2 のプログラム

〔プログラムに関する考察〕

二つの圧縮方法におけるデータ圧縮の効果について考える。

いま，同じ文字が n 個続く確率（出現率）を表 1 のとおり仮定する。例えば，配列 in の大きさが 100 の場合，1 割の 10 文字が連続しない一つの文字として存在する。また，4 割の 40 文字が 4 個連続する文字の割合である。このとき，4 個連続している文字列は 10 組となる。

圧縮後のデータの大きさを圧縮前のデータの大きさで割った値を圧縮比と定義すると，この表 1 の場合，〔圧縮方法その 1〕での圧縮比は [カ]，〔圧縮方法その 2〕での圧縮比は [キ] と算出できる。

表 1　同じ文字が n 個続く確率（出現率）

n	1	2	3	4
出現率	0.1	0.2	0.3	0.4

〔圧縮方法その 1〕の場合，圧縮対象のデータによっては，圧縮後のデータが圧縮前より大きくなってしまうことがある。<u>①最悪の場合</u>には，圧縮比は [ク] となってしまう。

設問 1　文字列“xyyyzzzzxxyzzzzz”の圧縮について，(1)，(2)に答えよ。

(1)　〔圧縮方法その 1〕によって圧縮した結果を答えよ。

(2)　〔圧縮方法その 2〕によって圧縮した結果を答えよ。

設問 2　図 1 中の [ア]，[イ] に入れる適切な字句を答えよ。

設問 3　図 2 中の [ウ] ～ [オ] に入れる適切な字句を答えよ。

設問 4　〔プログラムに関する考察〕について，(1)，(2)に答えよ。

(1)　[カ] ～ [ク] に入れる適切な数値を答えよ。

(2)　本文中の下線①とは，どのような場合か，20 字以内で答えよ。