3 次の**問 1〜3** に答えよ．

問 1 次の文章を読み，問い (1)〜(4) に答えよ．

1, 2, 3, 4, 5, 6 の数字をそれぞれ 1 回ずつ用いて，数字 6 桁の並びを作る．このようにして作った数字の並びを [123456] や [425316] のように書き，以後は「並び」と呼ぶことにする．同じ数字を 2 回以上使った [112345] などは考えない．並びの大小関係は，並びをそのまま 6 桁の整数とみなした場合の大小関係とする．例えば，[312465] と [321564] では，$312465 < 321564$ なので，[312465] の方が小さいと考える．並びの中で，最も小さいものから順に 4 番目までは，以下のようになる．

最も小さい並び	[123456]
2 番目に小さい並び	[123465]
3 番目に小さい並び	[123546]
4 番目に小さい並び	[123564]

(1) 最も大きい並びを答えよ．

(2) 4 番目に小さい並び [123564] に続く，5 番目から 7 番目に小さい並びを答えよ．

ある並び S があったとき，小さい順で考えて S の次にくる並びを「S の次の並び」のように呼ぶことにする．例えば，最も小さい並び [123456] の次の並びは，2 番目に小さい並び [123465] である．今，ある並びに対して，その次の並びを見つける手順について考える．

まず，ある並びと，その次の並びの間で，変化しない数字について考える．上に例示した並びを見ると，並びの左側が変化していないことに気付く．例えば，最も小さい並びと 2 番目に小さい並びでは，左側の 1234 の部分が変化していない．また，2 番目に小さい並びと 3 番目に小さい並びでは，左側の 123 の部分が変化していない．ここで，左から何桁目までが変化しないかは並びによって様々である．実は，変化しない数字については，次のような法則がある．

並びに含まれる数字を右から順に調べたとき，初めて右隣の数字よりも数が小さくなる桁があったとすると，その桁よりも左にある数字は変化しない．

並び [342651] を例として，この法則をあてはめてみる．[342651] の各桁を右から調べると，5 と 6 の各桁は右隣よりも大きいが，2 は右隣よりも小さい．よって，2 よりも左にある 34 の部分は変化しない．以後，「並びに含まれる数字を右から順に調べたとき，初めて右隣の数字よりも小さくなる桁」のことを「変化する部分の左端」と呼ぶことにする．[342651] の変化する部分の左端の数字は 2 である（図 1 (a) 参照）．

次に，変化する部分の左端の数字が，どの数字に変化するかを考える．[342651] の例では，左にある 34 の部分は変化しないので，651 のどれかの数字に変化する．ここで，もし変化する部分の左端が 1 に変化すると，元の [342651] よりも小さくなってしまう．そのように考えると，651 の中で 2 よりも大きい数のうち最小の数，すなわち 5 に変化するとわかる（図 1 (b) 参照）．

最後に，変化する部分の左端よりも右側の並びについては，残った数字を小さい順に並べればよい．[342651] の例では，すでに左から 3 桁が 345 であることはわかっているので，残った数字を小さい順に並べることにより，[342651] の次の並びは [345126] であるとわかる（図 1 (c) 参照）．

(3) 並び [436521] の変化する部分の左端の数字を答えよ．

(4) 並び [436521] の次の並びを答えよ．

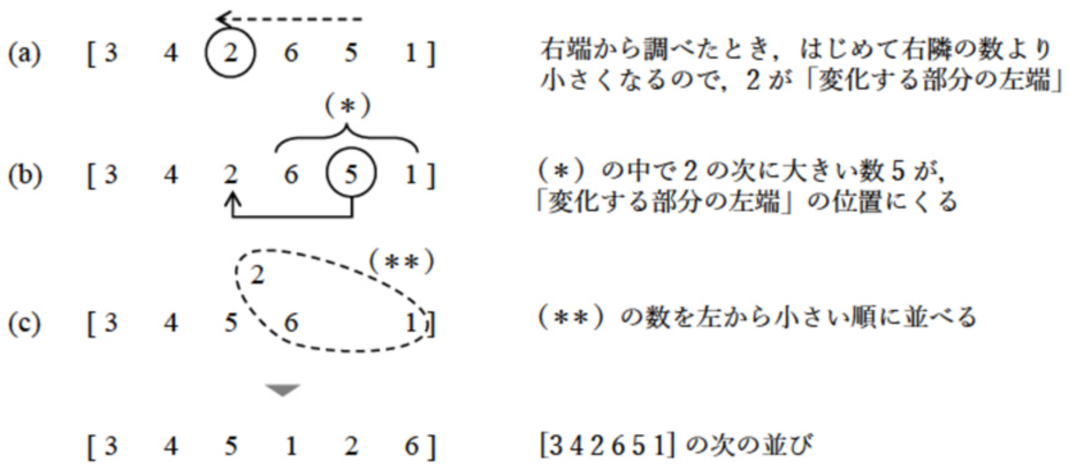

図 1　次の並びを求める手順

問 2 次の文章を読み，問い (1)・(2) に答えよ．

問 1 で考えた手順に基づいて，ある並びに対して，その次の並びを表示するプログラムを考える．ただし，最も大きい並びが与えられたときは，「最も大きい並びです」と表示する．並びは，配列 `Narabi` に入っているとする．また，配列の添字は 0 から始まるものとする．例えば，並び [342651] に対しては，図 2 のように値が格納される．

添字	0	1	2	3	4	5
`Narabi`	3	4	2	6	5	1

図 2　並びを格納する配列

このプログラムの中では，以下の関数を使用することができる（使わない関数があっても構わない）．

入れ換える (`i,j`)：`Narabi` の `i` 番目の要素と `j` 番目の要素を入れ換える．
降順にする (`i,j`)：`Narabi` の `i` 番目から `j` 番目の要素を降順に並べ替える．
逆順にする (`i,j`)：`Narabi` の `i` 番目から `j` 番目の要素の並び順を逆にする．

ただし，関数「**降順にする**」と関数「**逆順にする**」の引数 `i` と `j` は，`i` ≦ `j` となるように与えるものとし，`i` と `j` が等しい時には `Narabi` の内容は変化しないものとする．例として，`Narabi` が [3, 4, 2, 6, 5, 1] であったとき，それぞれの関数は以下のようにふるまう．

入れ換える (`1,4`) は，`Narabi` を [3, 5, 2, 6, 4, 1] に変える．
降順にする (`1,4`) は，`Narabi` を [3, 6, 5, 4, 2, 1] に変える．
逆順にする (`1,4`) は，`Narabi` を [3, 5, 6, 2, 4, 1] に変える．

このとき，以下の図 3 に示すプログラムにより，Narabi の次の並びを表示できる．ただし，「**要素数** (配列)」は，配列の要素数を返す関数である．また，「**表示する**」関数は，引数に配列を与えると，配列のすべての要素を順に表示するものとする．

```
ketasu = 要素数 (Narabi)
i = ketasu - 2
i >= 0 and [ ア ] の間繰り返す:
  └ i = i - 1
もし [ イ ] ならば:
  | j = ketasu - 1
  | i < j and [ ウ ] の間繰り返す:
  |   └ j = j - 1
  | [ エ ]([ オ ])
  | [ カ ]([ キ ])
  | 表示する (Narabi)
そうでなければ:
  └ 表示する ("最も大きい並びです")
```

図 3　次の並びを求めるプログラム

(1) 空欄 [ア] ～ [ウ] に入れるのに最も適当なものを次の選択肢から選び，その番号で答えよ．

選択肢：

① i >= 0　② i >= 1　③ i == 0　④ i == 1

⑤ Narabi[i-1] < Narabi[i]　⑥ Narabi[i-1] > Narabi[i]

⑦ Narabi[i] < Narabi[i+1]　⑧ Narabi[i] > Narabi[i+1]

⑨ Narabi[i] < Narabi[j]　⑩ Narabi[i] > Narabi[j]

⑪ Narabi[i-1] < Narabi[j]　⑫ Narabi[i-1] > Narabi[j]

(2) プログラムの 9 行目と 10 行目は，どちらも関数呼び出しである．空欄 $\boxed{\text{エ}}$・$\boxed{\text{カ}}$ に当てはまるものを関数の選択肢から，空欄 $\boxed{\text{オ}}$・$\boxed{\text{キ}}$ に当てはまるものを引数の選択肢から選び，その記号または番号で答えよ．

関数の選択肢：

ⓐ **入れ換える**　ⓑ **降順にする**　ⓒ **逆順にする**

引数の選択肢：

① `i, j`　② `i+1, j`　③ `i, j+1`

④ `i, ketasu-1`　⑤ `i+1, ketasu-1`

⑥ `j, ketasu-1`　⑦ `j+1, ketasu-1`

問 3 次の文章を読み，問い (1) に答えよ．

これまで，6 桁の数字はすべて違うという決まりで考えてきた．この問題では，同じ数字が含まれていてもよいという風に，決まりを緩める．すなわち，1, 2, 3, 4, 5, 6 の中から重複を許して 6 個の数字を選び，選んだ数字の並び替えを考える．ただし，いちど選んだ 6 個の数字は固定し，その 6 個の数字の並び替えのみを考えることとする．例えば，5 を 2 回用いて [123455] というような並びを作ることができる．同じ数字が含まれる場合も，並びの大小関係は，並びをそのまま 6 桁の整数とみなした場合の大小関係とする．従って，[123455] の次の並びは [123545] であり，[123545] の次の並びは [123554] である．

(1) このような同じ数字が含まれる場合には，図 3 に示した次の並びを求めるプログラムは，そのままでは正しく動作しない．しかし，空欄 $\boxed{\text{ア}}$・$\boxed{\text{ウ}}$ の条件式を変更することによって，同じ数字が含まれている場合にも正しく動作するようにできる．そのために，空欄 $\boxed{\text{ア}}$・$\boxed{\text{ウ}}$ に書くべき条件式を答えよ．