

問題 4 次のプログラムの説明および擬似言語の記述形式の説明を読み、設問に答えよ。

[プログラムの説明]

2次元配列中に格納されたデータを集計する関数 `syukei` である。

2次元配列には図 1 のように成績のデータが格納されている。

	科目 1	科目 2	科目 3
1 人目のデータ	80	100	70
2 人目のデータ	65	80	100
3 人目のデータ	45	30	80
4 人目のデータ	100	70	80
5 人目のデータ	70	90	80

図 1 5 人分 3 科目の成績データの例

集計作業は、個人ごとの合計を求め、合計の高い順に順位を出力し、最後に科目ごとの合計を出力する（図 2）。

なお、同点の場合は同順位とし、次に続く順位を欠番とする。

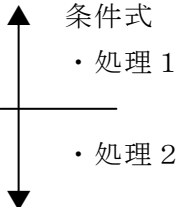
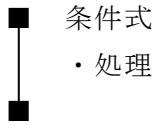
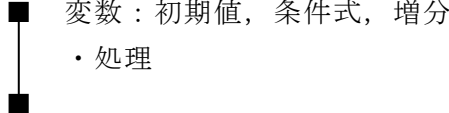
科目 1	科目 2	科目 3	合計	順位
80	100	70	250	1
65	80	100	245	3
45	30	80	155	5
100	70	80	250	1
70	90	80	240	4
360	370	410		

図 2 出力例

表 `syukei` の引数の仕様

変数名	入力／出力	意味
<code>n</code>	入力	人数（行数）
<code>k</code>	入力	科目数（列数）
<code>seiseki</code>	入力	データが格納された 2 次元配列

[擬似言語の記述形式の説明]

記述形式	説明
○	手続き，変数などの名前，型などを宣言する
・変数 ← 式	変数に式の値を代入する
/* 文 */	注釈を記述する
	<p>選択処理を示す。</p> <p>条件式が真の時は処理 1 を実行し，偽の時は処理 2 を実行する。</p>
	<p>前判定繰り返し処理を示す。</p> <p>条件式が真の間，処理を実行する。</p>
	<p>繰り返し処理を示す。</p> <p>変数に定数または変数で初期値が与えられ，条件式が成立する間処理を繰り返す。また，繰り返すごとに変数に増分が加えられる。</p>

なお，配列の要素位置は 0 から始まる。

[プログラム 1]

○syukei (整数型 : n, 整数型 : k, 整数型 : seiseki[n][k])

○整数型 : gokei[n], jyuni[n], kamoku_gokei[k]

○整数型 : i, j, cnt

/* 科目合計領域のゼロクリア */

```
■ i : 0, (1) , 1
  ・ kamoku_gokei[i] ← 0
■
```

/* 個人の合計, 科目ごとの合計を計算 */

```
■ i : 0, (2) , 1
  ・ gokei[i] ← 0
  ■ j : 0, j < k, 1
    ・ gokei[i] ← gokei[i] + seiseki[i][j]
    ・ (3)
  ■
■
```

/* i 番目のデータより大きいデータを数えて順位を求める */

```
■ i : 0, i < n, 1
  ・ cnt ← 1
  ■ j : 0, j < n, 1
    ▲ gokei[i] < gokei[j]
    ▼
    ・ cnt ← cnt + 1
  ■
  ・ jyuni[i] ← cnt
■
```

X

・ seiseki, gokei, jyuni の出力

・ kamoku_gokei の出力

<設問 1> プログラム 1 中の [] に入れるべき適切な字句を解答群から選べ。

(1), (2) の解答群

ア. $i < k$

ウ. $i < k + 1$

イ. $i < n$

エ. $i < n + 1$

(3) の解答群

ア. $\text{kamoku_gokei}[i] \leftarrow \text{kamoku_gokei}[i] + \text{seiseki}[i][j]$

イ. $\text{kamoku_gokei}[i] \leftarrow \text{kamoku_gokei}[i] + \text{seiseki}[j][i]$

ウ. $\text{kamoku_gokei}[j] \leftarrow \text{kamoku_gokei}[j] + \text{seiseki}[i][j]$

エ. $\text{kamoku_gokei}[j] \leftarrow \text{kamoku_gokei}[j] + \text{seiseki}[j][i]$

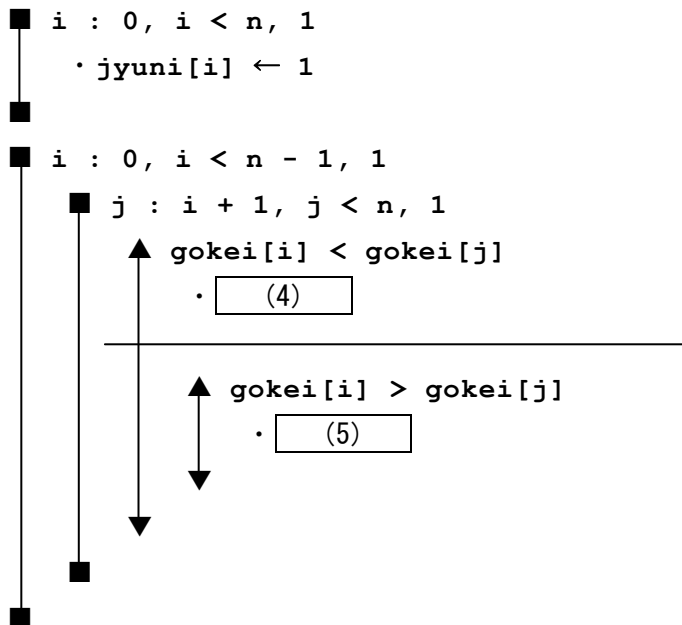
<設問 2> 次のプログラムの改良に関する記述を読み、プログラム 2 中の に
入るべき適切な字句を答えよ。

プログラム 1 の (X) の部分は、配列の行数の 2 乗の繰り返しになるため、関数 **syukei** に与える配列の行数が多いと処理時間が長くなる。

そこで、繰り返す回数を減らす方法を考える。

ここでは、**gokei[i]** と **gokei[j]** を比較したときに値が小さい方の添え字を使用して配列 **jyuni** の要素に 1 を加えることにした。

[プログラム 2]



(4) , (5) の解答群

- ア. **jyuni[i] ← jyuni[i] + 1**
ウ. **jyuni[j] ← jyuni[i] + 1**

- イ. **jyuni[i] ← jyuni[j] + 1**
エ. **jyuni[j] ← jyuni[j] + 1**