問2　スレッドを使用した並列実行（ソフトウェア）　(R1 秋-FE 午後問 2)

【解答】

［設問1］　a－オ，b－ウ
［設問2］　c－ウ，d－イ
［設問3］　e－エ

【解説】

スレッドを使用した並列実行に関する問題である。CPU（以下，プロセッサ）への割当ての単位をスレッドというが，一つのプログラムは複数のスレッドに分割され，プロセッサで実行される。プロセッサが一つであれば，順にスレッドを実行していくが，プロセッサが複数あるマルチプロセッサシステム（現在の MPU ではマルチコア）では，並列処理が可能なスレッドを並列実行することによって，実行時間の短縮を実現できる。

本問は，プログラムの高速化に関連して，スレッドを使用した並列実行における高速化率や，繰返しの処理における並列実行の可否が問われている。

設問 1 は計算ミスに注意する。設問 2 はプログラムによる配列処理を把握しなくてはならないのでトレースと検証が必要である。スレッド並列法を適用しない場合の処理を把握し，スレッド並列法を使用した場合との差異を吟味すればよい。

［設問1］

・空欄 a：スレッド並列法を適用した場合の高速化率 E は式 P のように示されている。式 P は，過去の情報処理技術者試験でも何度も出題されている，アムダールの法則そのものである。

$$E = \frac{1}{(1-r)+\frac{r}{n}} \qquad \cdots\cdots \text{式 P とする}$$

ここで，r $(0 \leqq r \leqq 1)$ である。

プログラム A のスレッドの個数 n は 2 で，このとき高速化率は $\frac{3}{2}$ になったので，次の式が成立する。

$$\frac{3}{2} = \frac{1}{(1-r)+\frac{r}{2}}$$

$$2 = 3\times(1-r)+3\times\frac{r}{2} = 3(1-r)+\frac{3r}{2} = 3-\frac{3r}{2}$$

$$1 = \frac{3r}{2}$$

$$r = \frac{2}{3}$$

したがって，正解は（オ）である。

・空欄 b：r が $\frac{3}{4}$ であるプログラム B で，スレッドの個数を増やした場合の高速化率の上限が問われている。

式 P において，$0 \leqq r \leqq 1$ だから，$\frac{r}{n}$ の値は n を 2，3，……と大きくしていくと反比例して小さくなっていく。スレッドの個数 n を非常に大きな値（∞）と仮定すると，$\frac{r}{n}$ の値は 0 に収束する。つまり，高速化率 E は，式 Q で近似される。

$$E = \frac{1}{(1-r)} \qquad \cdots\cdots \text{式 Q とする}$$

この式 Q に $r = \frac{3}{4}$ を代入すると，次のようになる。

$$E = \frac{1}{(1-\frac{3}{4})} = \frac{1}{\frac{1}{4}} = 4$$

したがって，正解は「4」の（ウ）である。

なお，式 P において n が 1，2 などと小さいときは，$(1-r)$ に 0 より大きい値の $\frac{r}{n}$ が加算されるため，E の値は式 Q の E の値より小さくなる。したがって，式 Q の E の値が高速化率の上限となる。

［設問2］

スレッド並列法を適用した場合に正しい結果が得られる場合と，正しい結果が得られない場合について考える。図 1 は「正しい結果が得られる場合の例」であり，図 2 は「正しい結果が得られない場合の二つの例」である。プログラムの破線の四角の処理をスレッドに分割した処理で実現した場合に，正しい結果が得られるか否かを見極めればよい。

図 1 のプログラム 1 の破線の四角の処理は，配列 b，c の要素 1～100 までの加算結果を配列 a の値とする処理である。

```
a[1]   ← b[1]   + c[1]
a[2]   ← b[2]   + c[2]
  :        :        :
a[50]  ← b[50]  + c[50]
a[51]  ← b[51]  + c[51]
a[52]  ← b[52]  + c[52]
  :        :        :
a[100] ← b[100] + c[100]
```

スレッド 1-1 で実現
制御変数 i_1 の取る範囲 1～50
まで繰り返す

スレッド 1-2 で実現
制御変数 i_2 の取る範囲 51～100
まで繰り返す

図 A　プログラム 1 の破線の四角の処理をスレッド 1-1 と 1-2 に分割

分割されたスレッド 1-1 と 1-2 はプログラム 1 の破線の四角の処理を制御変数（添字）1～50 と 51～100 の範囲に分けて行っているだけであり，それぞれの処理は制御変数による実行順序が異なっていても最終的には同じ結果が得られる。したがって，プログラム 1 は繰返しの範囲を分割したスレッド 1-1 と 1-2 に分けて，並列実行できる。

このように，独立して行える処理であれば問題はないが，操作の内容によっては一つの処理結果が他の処理結果に影響を及ぼすことがある。このようなときは正しい結果が保証されない。一方の処理結果が他の処理結果に影響を及ぼし，正しい結果が保証されない事象を，ここでは互いの処理が干渉すると定義して説明する。

・空欄 c：プログラム 2 の破線の四角の処理は次のとおりである。

```
i：1, i ≦ 100, 1
 ・a[i] ← a[i + 1] + b[i]
```

説明のために，配列 a，b に格納されているデータをそれぞれ a1～a101，b1～b101 とする（図 B）。

要素番号 →	1	2	…	50	51	…	100	101
配列 a	a1	a2	…	a50	a51	…	a100	a101

要素番号 →	1	2	…	50	51	…	100	101
配列 b	b1	b2	…	b50	b51	…	b100	b101

図 B　配列 a，b の内容

プログラム 2 の処理は図 C のようになる。スレッド 2-1 と 2-2 は制御変数 1～50 と 51～100 の範囲が分けられ，並列実行される。

```
a[1]   ← a[2]   + b[1]   = a2   + b1
a[2]   ← a[3]   + b[2]   = a3   + b2
  :        :        :            :
a[50]  ← a[51]  + b[50]  = a51  + b50
a[51]  ← a[52]  + b[51]  = a52  + b51
a[52]  ← a[53]  + b[52]  = a53  + b52
  :        :        :            :
a[100] ← a[101] + b[100] = a101 + b100
```

スレッド 2-1 で実現
制御変数 i_1 の取る範囲
1～50 まで繰り返す

スレッド 2-2 で実現
制御変数 i_2 の取る範囲
51～100 まで繰り返す

図 C　プログラム 2 の破線の四角の処理をスレッド 2-1 と 2-2 に分割

選択肢の内容から，網掛けの a[51]の値に着目する。

スレッド 2-1 と 2-2 は並列実行であるから，スレッド 2-1 で a[50]を求めた後にスレッド 2-2 が a[51]を求めるとは限らない。

a[51]に格納される値は a52＋b51 が正しいが，スレッド 2-2 がスレッド 2-1 より先に a[51]を求めた場合は，既に値 a52＋b51 が格納されている a[51]となる。その後，スレッド 2-1 で a[50]を求める処理を行った場合，正しい結果 a52＋b51 とならない。これは，a[51]の値が，既にスレッド 2-2 で a52＋b51 に更新されているためである。したがって，「a[51]の値（a51）をスレッド 2-1 で参照するより先にスレッド 2-2 で更新（a52＋b51）する」ことがあり，（ウ）が正解である。

```
a[50] ← a[51] + b[50] = a52＋b51 + b50
```

なお，他の選択肢の誤りは次のとおりである。図 C を確認してほしい。
ア，イ：スレッド 2-1 は a[50]までの更新処理のため，「a[51]の値をスレッド 2-1 で更新する」ことはない。
エ：スレッド 2-2 は a[51]の値を参照することはない。

・空欄 d：プログラム 3 の破線の四角の処理は次のとおりである。

```
i：2, i ≦ 101, 1
 ・a[i] ← a[i − 1] + b[i]
```

なお，配列 a，b に格納されているデータは図 B と同じとする。スレッド 3-1，3-2 の処理は次のようになる。

```
a[2]   ← a[1]   + b[2]   = a1 + b2
a[3]   ← a[2]   + b[3]   = a1 + b2 + b3
  :        :               :
a[50]  ← a[49]  + b[50]  = a1 + b2 + b3 + … + b50
a[51]  ← a[50]  + b[51]  = a1 + b2 + b3 + … + b51
a[52]  ← a[51]  + b[52]  = a1 + b2 + b3 + … + b52
  :        :               :
a[100] ← a[99]  + b[100] = a1 + b2 + b3 + … + b100
a[101] ← a[100] + b[101] = a1 + b2 + b3 + … + b101
```

スレッド 3-1 で実現
制御変数 i_1 の取る範囲を 2～51 まで繰り返す

スレッド 3-2 で実現
制御変数 i_2 の取る範囲を 52～101 まで繰り返す

図 D　プログラム 3 の破線の四角の処理をスレッド 3-1 と 3-2 に分割

本来の a[52]は a[51]を求めた後に求めるので，「a1＋b2＋b3＋ …… ＋b52」とならなければならない。しかし，この処理をスレッド 3-1 と 3-2 に分割し，並列実行すると，a[51]を求めた後に a[52]を求める保証はない。スレッド 3-2 で先に a[52]を求めてしまうと a[51]には値 a51 が格納されているため a51＋b52 となってしまい，正しい結果「a1＋b2＋b3＋ … ＋b52」とならない。

したがって，「a[51]の値をスレッド 3-1 で更新するより先にスレッド 3-2 で参照（参照値は a51）する」ことがあり，正しい結果が保証されない（イ）が正解である。

なお，他の選択肢の誤りは次のとおりである。図 D を確認してほしい。
ア：スレッド 3-2 は a[51]の値を参照するが，a[51]の値を更新しない。
ウ，エ：スレッド 3-1 は a[51]の値を参照することはない。

［設問3］

スレッド並列法を適用できる場合とできない場合の配列 ip の要素の値を考察する。プログラム 4 の破線の四角の処理をスレッド 4-1 で i＝1～5 まで実行すると，ip[i]は 1～5 なので，配列 a[1]～a[5]は図 E の＜更新後の配列 a＞のようになる。なお，配列 a，b の要素数は 10 とする。

＜更新前の配列 a，b，ip＞

要素番号 →	1	2	3	4	5	6	7	8	9	10
配列 a	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10

要素番号 →	1	2	3	4	5	6	7	8	9	10
配列 b	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10

要素番号 →	1	2	3	4	5	6	7	8	9	10
配列 ip	1	2	3	4	5	e				

＜更新後の配列 a＞

スレッド 4-1 による i＝1～5 までの処理

i	b[i]	ip[i]	a[ip[i]] ← b[i]
1	b[1]＝b1	ip[1]＝1	a[1] ← b1
2	b[2]＝b2	ip[2]＝2	a[2] ← b2
:	:	:	:
5	b[5]＝b5	ip[5]＝5	a[5] ← b5

要素番号 →	1	2	3	4	5	6	7	8	9	10
配列 a	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10

図 E　更新前の配列 a，b，ip と更新後の配列 a

スレッド 4-1 と 4-2 は並列実行が可能なので，互いの処理が干渉することがなければ正しい結果が得られるはずである。このことは設問 2 によって確認できた。しかし，いったん更新した内容を再更新したり，正しい参照データが得られなかったりする場合は互いの処理が干渉し，正しい結果とならないこともある。このことを踏まえて，選択肢を確認していくと次のようになる。

ア：ip[6]の値が 1 の場合，スレッド 4-1 の i_1＝1 で更新された a[1]の値 b1 が b[6]＝b6 で更新される可能性があり，値 b1 が保証されない。このことは ip[7]の値が 2 でもいえることであり，正しい結果の保証がない（誤り）。

イ：ip[6]の値が 5 であり，スレッド 4-1 の i_1＝5 と干渉しているため，a[5]の値が b[6]＝b6 で更新される可能性があり，正しい結果の保証がない（誤り）。

ウ：ip[8]の値が 4 であり，スレッド 4-1 の i_1＝4 と干渉しているため，a[4]の値が a[8]＝b8 で更新される可能性があり，正しい結果の保証がない（誤り）。

エ：ip[6]～ip[10]の値が 6～10 の値であり，スレッド 4-1 の i_1＝1～5 に ip[1]～ip[5]の値 1～5 と干渉していない。6～10 の値は排他的な値であるため，配列 a の a[6]～a[10]にはスレッド 4-2 の i＝6～10 で格納される b6～b10 の値が保証される。

以上から，正解は（エ）である。