問8　空き領域の管理（データ構造及びアルゴリズム）　（H26春-FE 午後問8）

【解答】
［設問1］　a－ア，b－エ
［設問2］　c－イ，d－ウ，e－ウ
［設問3］　f－ア，g－エ，h－イ

【解説】
領域中のセルについて，割当てと解放の処理を行うアルゴリズムの問題である。設問1，設問2は関数 **Free**，関数 **Alloc** の空きリスト更新処理に関する問題，設問3はアルゴリズムを考察する問題である。空きリストの形式や関数 **Alloc**，関数 **Free** の説明をよく読んで理解し，セルと空きリストの状態を具体的な図にしながら考えると解答は導きやすい。ただし，空きリストの扱いと解答が終わるまでの時間を考慮すると，難易度はやや難しいといえる。

はじめに，空きリストの形式及びセルの割当てと解放の処理について確認する。

各セルには，セル位置を指定するための連続する整数が対応している。各セルは，“空き”又は“割当済み”のいずれかの状態にあり，領域中のどのセルが“空き”の状態にあるかという情報を，空きリストとして保持している。空きリストは，次のように，一つの連続した“空き”セルの始点と終点の組で表す。

{{始点$_1$，終点$_1$}，{始点$_2$，終点$_2$}，…，{始点$_N$，終点$_N$}}

領域の初期状態は，全てのセルが空いており，空きリストは {{−∞，＋∞}} で表す。この場合，空きリストは一つの連続した領域となり，空きリスト中の組の個数は1となる。なお，“−∞”は，領域中のどのセルの位置の値よりも小さい整数を，“＋∞”は，領域中のどのセルの位置の値よりも大きい整数を表す記号である。

領域の初期状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1									終点 1	

注記　セルは0から9までの10個と仮定する

セルの割当て処理は，関数 **Alloc**（始点，終点）で行う。**Alloc** は引数で指定した始点から終点までの連続した“空き”セルを“割当済み”として，空きリストから取り除く。例えば，上記の初期状態から，**Alloc**（1，2）を実行すると，1と2のセルが“割当済み”となり，空きリストから取り除かれる。このため，−∞から＋∞までの連続していたセルは，−∞から0までと3から＋∞までの2組に分かれる。よって，実行後の空きリストは {{−∞，0}，{3，＋∞}} となる。

Alloc（1，2）を実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1	終点 1		始点 2						終点 2	

注記　網掛け部分は“割当済み”の状態を表す

続いて，**Alloc**（6，8）を実行すると，6から8のセルが“割当済み”となり，空きリストから取り除かれる。このため，3から＋∞までの連続していたセルは，3から5までと9から＋∞までの2組に分かれる。よって，実行後の空きリストは，{{−∞，0}，{3，5}，{9，＋∞}} の3組になる。

Alloc（6，8）を実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1	終点 1		始点 2		終点 2				始点 3	終点 3

プログラム中では，空きリスト {{始点$_1$，終点$_1$}，{始点$_2$，終点$_2$}，…，{始点$_N$，終点$_N$}} の始点$_i$（i：1，2，…，N）の値は，配列「始点」に，終点$_i$（i：1，2，…，N）の値は，配列「終点」に格納されている。**Alloc**（6，8）を実行後の二つの配列は次のようになり，空きリスト中の組の個数を表すNは3となる。

	1	2	3	4	5	…
始点	−∞	3	9			
終点	0	5	＋∞			

セルの解放処理は，関数 **Free**（始点，終点）で行う。**Free** は引数で指定した始点から終点までの連続した“割当済み”セルを“空き”として，空きリストに戻す。例えば，**Free**（6，7）を実行すると，6と7のセルが“空き”として，空きリストに戻される。このため，3から5までの“空き”セルと解放した6，7のセルがつながって，一つの連続した“空き”セルの組 {3，7} となる。よって，実行後の空きリストは，{{−∞，0}，{3，7}，{9，＋∞}} の3組になる。

Free（6，7）を実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1	終点 1		始点 2				終点 2		始点 3	終点 3

［設問1］
関数 **Free** の空きリスト更新処理に関する問題である。

・空欄a：引数の状況が「終点$_i$＝始点$_p$−1かつ終点$_p$＋1＝始点$_{i+1}$」の場合である。
分かりやすくするために，次の図を使って考える。

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i	終点 i			始点 i+1	終点 i+1		

空きリストの二つの組 {始点$_i$，終点$_i$} と {始点$_{i+1}$，終点$_{i+1}$} をそれぞれ {3，4} と {7，8} とすると，終点$_i$は4，始点$_{i+1}$は7となる。終点$_i$＝始点$_p$−1なので，始点$_p$は1を引いた値が4になる5である。また，終点$_p$＋1＝始点$_{i+1}$なので，終点$_p$は1を足した値が7になる6である。よって，**Free**（5，6）を実行し，5と6のセルを解放することになる。これによって，3から8までが一つの連続した“空き”セルの組 {3，8} になる。これを始点，終点の関係で表すと，{始点$_i$，終点$_i$} と {始点$_{i+1}$，終点$_{i+1}$} の二つの組は {始点$_i$，終点$_{i+1}$} の一つの組に置き換わる。したがって，空欄aは（ア）が正解である。

実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i					終点 i+1		

・空欄b：引数の状況が「終点$_i$＜始点$_p$−1かつ終点$_p$＋1＜始点$_{i+1}$」の場合である。
分かりやすくするために，次の図を使って考える。

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i	終点 i				始点 i+1	終点 i+1	

空きリストの二つの組 {始点$_i$，終点$_i$} と {始点$_{i+1}$，終点$_{i+1}$} をそれぞれ {3，4} と {9，＋∞} とすると，終点$_i$は4，始点$_{i+1}$は9となる。終点$_i$＜始点$_p$−1かつ終点$_p$＋1＜始点$_{i+1}$なので，始点$_p$の値から1を引いた値が4よりも大きく，終点$_p$の値に1を足した値が9よりも小さいことになる。このことから，1を引いて5，1を足して8になる始点$_p$と終点$_p$は，それぞれ6と7になる。**Free**（6，7）を実行し，6と7のセルを解放するので，新たに {6，7} の組が“空き”セルとして空きリストに加わる。そして，そのために {3，4} と {9，＋∞} の間に {6，7} の組を挿入する。これを始点，終点の関係で表すと，組 {始点$_i$，終点$_i$} の直後に {始点$_p$，終点$_p$} を挿入することになる。したがって，空欄bは（エ）が正解である。

実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i	終点 i		新たな組		始点 i+1	終点 i+1	

［設問2］
関数 **Alloc** の空きリスト更新処理に関するプログラムの穴埋め問題である。

［プログラム］

```
○関数：Alloc(整数型：始点P，整数型：終点P)
○整数型：I，L
01  ・I ← 1
02  ■ 終点P ＞ 終点[I]                          ┐
03    ・I ← I＋1                                 │Allocの説明(1)
04  ■                                            ┘
05  ▲ 始点[I] ≦ 始点P                           ┐
06    ▲ (始点[I] ＝ 始点P) and (終点P ＝ 終点[I])  ┐
07      ■ L：I＋1，L ≦ N，1                       │
08        ・始点[L−1] ← 始点[L]                   │表1
09        ・終点[L−1] ← 終点[L]                   │引数の状況の1番目
10      ■                                         │
11      ・N ← N−1                                 ┘
12
13    ▲(始点[I] ＝ 始点P) and (終点P ＜ 終点[I])  ┐表1
14      ・ [ c ]                                  ┘引数の状況の2番目
15
16    ▲(始点[I] ＜ 始点P) and (終点P ＝ 終点[I])  ┐表1
17      ・ [ d ]                                  ┘引数の状況の3番目
18
19    ▲(始点[I] ＜ 始点P) and (終点P ＜ 終点[I])  ┐
20      ■ L： [ e ]                               │
21        ・始点[L＋1] ← 始点[L]                  │
22        ・終点[L＋1] ← 終点[L]                  │表1
23      ■                                         │引数の状況の4番目
24      ・始点[I＋1] ← 終点P＋1                   │
25      ・終点[I＋1] ← 終点[I]                    │
26      ・終点[I] ← 始点P−1                       │
27      ・N ← N＋1                                ┘
28
29  ・print("一部又は全体が割当済み")      /* " " 内の文字列を表示*/
30
```

行番号02〜04：
割当てを行うセルが含まれる可能性がある組を探す処理。次の行番号05の条件式と合わせて〔関数 **Alloc** の説明〕(1)に該当する。

行番号06〜12：
割当てを行うセルと“空き”セルの組の一つが全く同じ領域の場合の処理（表1の引数の状況の1番目）

行番号13〜15：
割当てを行うセルと“空き”セルの組の始点が同じで，割当てを行うセルの終点がその“空き”セルの組の範囲内にある場合の処理（表1の引数の状況の2番目）

行番号16〜18：
割当てを行うセルの始点が，ある“空き”セルの組の範囲内にあり，割当てを行うセルの終点がその“空き”セルの組の終点と同じ場合の処理（表1の引数の状況の3番目）

行番号19〜28：
割当てを行うセルが，ある“空き”セルの組の範囲内にある場合の処理（表1の引数の状況の4番目）

行番号29：
割り当てるセルが存在しない場合の処理

・空欄c：条件式「**(始点[I]＝始点P) and (終点P＜終点[I])**」は，表1の引数の状況の2番目に対応する。分かりやすくするために，次の図を使って考える。

−∞	0	1	2	3	4	5	6	7	8	9	＋∞

{始点[I]，終点[I]} を {3，6} とすると，始点[I]＝始点P なので，始点Pは3，終点P＜終点[I]なので，終点Pは6よりも小さい値となる。仮に，{始点P，終点P} を {3，4} と考えると，3と4のセルが“割当済み”になり，空きリストから除外される。このため，“空き”セルの組 {3，6} は，{5，6} に更新される。この場合，終点は変わらないので，始点だけを更新する。

問8　空き領域の管理（データ構造及びアルゴリズム）　（H26春-FE 午後問8）

【解答】
［設問1］　a－ア，b－エ
［設問2］　c－イ，d－ウ，e－ウ
［設問3］　f－ア，g－エ，h－イ

【解説】
領域中のセルについて，割当てと解放の処理を行うアルゴリズムの問題である。設問1，設問2は関数 **Free**，関数 **Alloc** の空きリスト更新処理に関する問題，設問3はアルゴリズムを考察する問題である。空きリストの形式や関数 **Alloc**，関数 **Free** の説明をよく読んで理解し，セルと空きリストの状態を具体的な図にしながら考えると解答は導きやすい。ただし，空きリストの扱いと解答が終わるまでの時間を考慮すると，難易度はやや難しいといえる。

はじめに，空きリストの形式及びセルの割当てと解放の処理について確認する。

各セルには，セル位置を指定するための連続する整数が対応している。各セルは，“空き”又は“割当済み”のいずれかの状態にあり，領域中のどのセルが“空き”の状態にあるかという情報を，空きリストとして保持している。空きリストは，次のように，一つの連続した“空き”セルの始点と終点の組で表す。

{{始点$_1$，終点$_1$}，{始点$_2$，終点$_2$}，…，{始点$_N$，終点$_N$}}

領域の初期状態は，全てのセルが空いており，空きリストは {{−∞，＋∞}} で表す。この場合，空きリストは一つの連続した領域となり，空きリスト中の組の個数は1となる。なお，“−∞”は，領域中のどのセルの位置の値よりも小さい整数を，“＋∞”は，領域中のどのセルの位置の値よりも大きい整数を表す記号である。

領域の初期状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1									終点 1	

注記　セルは0から9までの10個と仮定する

セルの割当て処理は，関数 **Alloc**（始点，終点）で行う。**Alloc** は引数で指定した始点から終点までの連続した“空き”セルを“割当済み”として，空きリストから取り除く。例えば，上記の初期状態から，**Alloc**（1，2）を実行すると，1と2のセルが“割当済み”となり，空きリストから取り除かれる。このため，−∞から＋∞までの連続していたセルは，−∞から0までと3から＋∞までの2組に分かれる。よって，実行後の空きリストは {{−∞，0}，{3，＋∞}} となる。

Alloc（1，2）を実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1	終点 1		始点 2						終点 2	

注記　網掛け部分は“割当済み”の状態を表す

続いて，**Alloc**（6，8）を実行すると，6から8のセルが“割当済み”となり，空きリストから取り除かれる。このため，3から＋∞までの連続していたセルは，3から5までと9から＋∞までの2組に分かれる。よって，実行後の空きリストは，{{−∞，0}，{3，5}，{9，＋∞}} の3組になる。

Alloc（6，8）を実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1	終点 1		始点 2		終点 2				始点 3	終点 3

プログラム中では，空きリスト {{始点$_1$，終点$_1$}，{始点$_2$，終点$_2$}，…，{始点$_N$，終点$_N$}} の始点$_i$（i：1，2，…，N）の値は，配列「始点」に，終点$_i$（i：1，2，…，N）の値は，配列「終点」に格納されている。**Alloc**（6，8）を実行後の二つの配列は次のようになり，空きリスト中の組の個数を表すNは3となる。

	1	2	3	4	5	…
始点	−∞	3	9			
終点	0	5	＋∞			

セルの解放処理は，関数 **Free**（始点，終点）で行う。**Free** は引数で指定した始点から終点までの連続した“割当済み”セルを“空き”として，空きリストに戻す。例えば，**Free**（6，7）を実行すると，6と7のセルが“空き”として，空きリストに戻される。このため，3から5までの“空き”セルと解放した6，7のセルがつながって，一つの連続した“空き”セルの組 {3，7} となる。よって，実行後の空きリストは，{{−∞，0}，{3，7}，{9，＋∞}} の3組になる。

Free（6，7）を実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
	始点 1	終点 1		始点 2				終点 2		始点 3	終点 3

［設問1］
関数 **Free** の空きリスト更新処理に関する問題である。

・空欄a：引数の状況が「終点$_i$＝始点$_p$−1かつ終点$_p$＋1＝始点$_{i+1}$」の場合である。
分かりやすくするために，次の図を使って考える。

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i	終点 i			始点 i+1	終点 i+1		

空きリストの二つの組 {始点$_i$，終点$_i$} と {始点$_{i+1}$，終点$_{i+1}$} をそれぞれ {3，4} と {7，8} とすると，終点$_i$は4，始点$_{i+1}$は7となる。終点$_i$＝始点$_p$−1なので，始点$_p$は1を引いた値が4になる5である。また，終点$_p$＋1＝始点$_{i+1}$なので，終点$_p$は1を足した値が7になる6である。よって，**Free**（5，6）を実行し，5と6のセルを解放することになる。これによって，3から8までが一つの連続した“空き”セルの組 {3，8} になる。これを始点，終点の関係で表すと，{始点$_i$，終点$_i$} と {始点$_{i+1}$，終点$_{i+1}$} の二つの組は {始点$_i$，終点$_{i+1}$} の一つの組に置き換わる。したがって，空欄aは（ア）が正解である。

実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i					終点 i+1		

・空欄b：引数の状況が「終点$_i$＜始点$_p$−1かつ終点$_p$＋1＜始点$_{i+1}$」の場合である。
分かりやすくするために，次の図を使って考える。

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i	終点 i				始点 i+1	終点 i+1	

空きリストの二つの組 {始点$_i$，終点$_i$} と {始点$_{i+1}$，終点$_{i+1}$} をそれぞれ {3，4} と {9，＋∞} とすると，終点$_i$は4，始点$_{i+1}$は9となる。終点$_i$＜始点$_p$−1かつ終点$_p$＋1＜始点$_{i+1}$なので，始点$_p$の値から1を引いた値が4よりも大きく，終点$_p$の値に1を足した値が9よりも小さいことになる。このことから，1を引いて5，1を足して8になる始点$_p$と終点$_p$は，それぞれ6と7になる。**Free**（6，7）を実行し，6と7のセルを解放するので，新たに {6，7} の組が“空き”セルとして空きリストに加わる。そして，そのために {3，4} と {9，＋∞} の間に {6，7} の組を挿入する。これを始点，終点の関係で表すと，組 {始点$_i$，終点$_i$} の直後に {始点$_p$，終点$_p$} を挿入することになる。したがって，空欄bは（エ）が正解である。

実行後の状態

−∞	0	1	2	3	4	5	6	7	8	9	＋∞
				始点 i	終点 i		新たな組		始点 i+1	終点 i+1	

［設問2］
関数 **Alloc** の空きリスト更新処理に関するプログラムの穴埋め問題である。

［プログラム］

```
○関数：Alloc(整数型：始点P，整数型：終点P)
○整数型：I，L
01  ・I ← 1
02  ■ 終点P ＞ 終点[I]                          ┐
03    ・I ← I＋1                                 │Allocの説明(1)
04  ■                                            ┘
05  ▲ 始点[I] ≦ 始点P                           ┐
06    ▲ (始点[I] ＝ 始点P) and (終点P ＝ 終点[I])  ┐
07      ■ L：I＋1，L ≦ N，1                       │
08        ・始点[L−1] ← 始点[L]                   │表1
09        ・終点[L−1] ← 終点[L]                   │引数の状況の1番目
10      ■                                         │
11      ・N ← N−1                                 ┘
12
13    ▲(始点[I] ＝ 始点P) and (終点P ＜ 終点[I])  ┐表1
14      ・ [ c ]                                  ┘引数の状況の2番目
15
16    ▲(始点[I] ＜ 始点P) and (終点P ＝ 終点[I])  ┐表1
17      ・ [ d ]                                  ┘引数の状況の3番目
18
19    ▲(始点[I] ＜ 始点P) and (終点P ＜ 終点[I])  ┐
20      ■ L： [ e ]                               │
21        ・始点[L＋1] ← 始点[L]                  │
22        ・終点[L＋1] ← 終点[L]                  │表1
23      ■                                         │引数の状況の4番目
24      ・始点[I＋1] ← 終点P＋1                   │
25      ・終点[I＋1] ← 終点[I]                    │
26      ・終点[I] ← 始点P−1                       │
27      ・N ← N＋1                                ┘
28
29  ・print("一部又は全体が割当済み")      /* " " 内の文字列を表示*/
30
```

行番号02〜04：
割当てを行うセルが含まれる可能性がある組を探す処理。次の行番号05の条件式と合わせて〔関数 **Alloc** の説明〕(1)に該当する。

行番号06〜12：
割当てを行うセルと“空き”セルの組の一つが全く同じ領域の場合の処理（表1の引数の状況の1番目）

行番号13〜15：
割当てを行うセルと“空き”セルの組の始点が同じで，割当てを行うセルの終点がその“空き”セルの組の範囲内にある場合の処理（表1の引数の状況の2番目）

行番号16〜18：
割当てを行うセルの始点が，ある“空き”セルの組の範囲内にあり，割当てを行うセルの終点がその“空き”セルの組の終点と同じ場合の処理（表1の引数の状況の3番目）

行番号19〜28：
割当てを行うセルが，ある“空き”セルの組の範囲内にある場合の処理（表1の引数の状況の4番目）

行番号29：
割り当てるセルが存在しない場合の処理

・空欄c：条件式「**(始点[I]＝始点P) and (終点P＜終点[I])**」は，表1の引数の状況の2番目に対応する。分かりやすくするために，次の図を使って考える。

−∞	0	1	2	3	4	5	6	7	8	9	＋∞

{始点[I]，終点[I]} を {3，6} とすると，始点[I]＝始点P なので，始点Pは3，終点P＜終点[I]なので，終点Pは6よりも小さい値となる。仮に，{始点P，終点P} を {3，4} と考えると，3と4のセルが“割当済み”になり，空きリストから除外される。このため，“空き”セルの組 {3，6} は，{5，6} に更新される。この場合，終点は変わらないので，始点だけを更新する。