次の**問8**は必須問題です。必ず解答してください。

問8　次のアルゴリズムの説明を読んで，設問1～3に答えよ。

セルを1列に連続して並べた領域がある。この領域中のセルについて，割当てと解放の処理を行う。

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。領域のセル数や，対応する整数の範囲には，特に制限がない。

各セルは，“空き”又は“割当済み”のいずれかの状態にある。現在，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。

関数 Alloc(始点，終点) 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。関数 Free(始点，終点) 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。

〔空きリストの説明〕

空きリストの形式を，次に示す。

$\{\{始点_1，終点_1\}，\{始点_2，終点_2\}，\cdots，\{始点_N，終点_N\}\}$

$\{始点_i，終点_i\}$（$始点_i \leqq 終点_i$）は，一つの連続した“空き”セルの先頭位置と終端位置の組（以下，組という）で，$始点_1 < 始点_2 < \cdots < 始点_N$ である。

割当て・解放の処理と空きリストの状態の例を，次の (1) ～ (3) に示す。ここで，セル □ 中の数字は，セル位置を表す。また，□ は“空き”を，▨ は“割当済み”を，それぞれ表す。

(1)　領域の初期状態は，全セルが空いている。空きリストは {{ -∞，+∞ }} で表す。

…	0	1	2	3	4	5	6	7	8	9	…

初期状態の空きリスト：{{ -∞，+∞ }}

ここで，記号“-∞”は，領域中のどのセル位置の値よりも小さい整数を表し，記号“+∞”は，領域中のどのセル位置の値よりも大きい整数を表すものとする。また，セル位置 -∞ ～ +∞ のうち，領域外の部分には“空き”セルが並んでいるもの

とする。

(2) 関数 Alloc で“割当済み”としたセルは，空きリストから取り除く。例えば，(1) の初期状態から，Alloc(1, 2) と Alloc(6, 8) を実行すると，次のようになる。

…	0	1	2	3	4	5	6	7	8	9	…

実行後の空きリスト：{{ -∞, 0}, {3, 5}, {9, +∞}}

実行後，空きリスト中の組の個数は3となる。

(3) 関数 Free で解放したセルは，空きリストに戻す。例えば，(2) の実行後の状態から，Free(6, 7) を実行すると，次のようになる。

…	0	1	2	3	4	5	6	7	8	9	…

実行後の空きリスト：{{ -∞, 0}, {3, 7}, {9, +∞}}

実行後，解放された“空き”セルの組 { 6, 7 } は，実行前の“空き”セルの組 { 3, 5 } とつながって一つの連続した“空き”セルの組 { 3, 7 } となるので，空きリスト中の組の個数は3となる。

〔関数 Alloc の説明〕

関数 Alloc(始点$_p$, 終点$_p$) の処理手順は，次のとおりである。

なお，引数の値は，$-\infty <$ 始点$_p \leqq$ 終点$_p < +\infty$ を満たしているものとする。

(1) 空きリスト中に，始点$_i \leqq$ 始点$_p$ かつ 終点$_p \leqq$ 終点$_i$ を満たす組 { 始点$_i$, 終点$_i$ } が存在すれば (2) へ進む。存在しなければ，“一部又は全体が割当済み”を表示して，処理を終了する。

(2) 割当てが可能であるので，表 1 に従って，引数の状況に対応した空きリストの更新処理を実行して，処理を終了する。

表1　関数 Alloc の空きリスト更新処理

引数の状況	空きリストの更新処理
始点$_i$ ＝ 始点$_p$ かつ 終点$_p$ ＝ 終点$_i$	組 { 始点$_i$, 終点$_i$ } を取り除く。
始点$_i$ ＝ 始点$_p$ かつ 終点$_p$ ＜ 終点$_i$	組 { 始点$_i$, 終点$_i$ } を組 [　　] で置き換える。
始点$_i$ ＜ 始点$_p$ かつ 終点$_p$ ＝ 終点$_i$	組 { 始点$_i$, 終点$_i$ } を組 [　　] で置き換える。
始点$_i$ ＜ 始点$_p$ かつ 終点$_p$ ＜ 終点$_i$	組 { 始点$_i$, 終点$_i$ } を二つの組 { 始点$_i$, 始点$_p$ −1 } と { 終点$_p$ +1, 終点$_i$ } で置き換える。

注記　網掛けの部分は表示していない。

〔関数 Free の説明〕

関数 Free(始点$_p$, 終点$_p$) の処理手順は，次のとおりである。

なお，引数の値は，$-\infty <$ 始点$_p \leqq$ 終点$_p < +\infty$ を満たしているものとする。

(1)　空きリスト中に，終点$_i <$ 始点$_p$ かつ 終点$_p <$ 始点$_{i+1}$ を満たす連続する二つの組 { 始点$_i$, 終点$_i$ } と { 始点$_{i+1}$, 終点$_{i+1}$ } が存在すれば (2) へ進む。存在しなければ，“一部又は全体が割当済みでない”を表示して，処理を終了する。

(2)　解放が可能であるので，表 2 に従って，引数の状況に対応した空きリストの更新処理を実行して，処理を終了する。

表2　関数 Free の空きリスト更新処理

引数の状況	空きリストの更新処理
終点$_i$ ＝ 始点$_p$ −1 かつ 終点$_p$ +1 ＝ 始点$_{i+1}$	二つの組 { 始点$_i$, 終点$_i$ } と { 始点$_{i+1}$, 終点$_{i+1}$ } を一つの組 [a] で置き換える。
終点$_i$ ＝ 始点$_p$ −1 かつ 終点$_p$ +1 ＜ 始点$_{i+1}$	組 { 始点$_i$, 終点$_i$ } を組 { 始点$_i$, 終点$_p$ } で置き換える。
終点$_i$ ＜ 始点$_p$ −1 かつ 終点$_p$ +1 ＝ 始点$_{i+1}$	組 { 始点$_{i+1}$, 終点$_{i+1}$ } を組 { 始点$_p$, 終点$_{i+1}$ } で置き換える。
終点$_i$ ＜ 始点$_p$ −1 かつ 終点$_p$ +1 ＜ 始点$_{i+1}$	組 { 始点$_i$, 終点$_i$ } の直後に組 [b] を挿入する。

設問1　本文中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　{始点$_i$，終点$_{i+1}$}　　イ　{始点$_i$，終点$_p$}

ウ　{始点$_p$，終点$_{i+1}$}　　エ　{始点$_p$，終点$_p$}

設問2　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

関数 Alloc の説明に基づいて，プログラムを作成した。

空きリスト中の現在の組の個数は大域整数型変数 N に格納されている。空きリスト{{始点$_1$，終点$_1$}，{始点$_2$，終点$_2$}，…，{始点$_N$，終点$_N$}}については，始点$_i$（i：1，2，…，N）の値は大域整数型配列 **始点** の要素 **始点**[i] に，終点$_i$（i：1，2，…，N）の値は大域整数型配列 **終点** の要素 **終点**[i] に，それぞれ格納されている。これらの配列は，十分に大きいものとする。

〔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

```
• I ← 1
■ 終点P ＞ 終点[I]
  • I ← I+1
■
▲ 始点[I] ≦ 始点P
  ▲ (始点[I] ＝ 始点P) and (終点P ＝ 終点[I])
    ■ L: I+1, L ≦ N, 1
      • 始点[L−1] ← 始点[L]
      • 終点[L−1] ← 終点[L]
    ■
    • N ← N−1
  ▲ (始点[I] ＝ 始点P) and (終点P ＜ 終点[I])
    • [        c        ]
  ▲ (始点[I] ＜ 始点P) and (終点P ＝ 終点[I])
    • [        d        ]
  ▲ (始点[I] ＜ 始点P) and (終点P ＜ 終点[I])
    ■ L: [        e        ]
      • 始点[L+1] ← 始点[L]
      • 終点[L+1] ← 終点[L]
    ■
    • 始点[I+1] ← 終点P+1
    • 終点[I+1] ← 終点[I]
    • 終点[I] ← 始点P−1
    • N ← N+1
  • print("一部又は全体が割当済み")   /* "  " 内の文字列を表示 */
```

c, dに関する解答群

ア 始点[I] ← 始点P−1	イ 始点[I] ← 終点P+1
ウ 終点[I] ← 始点P−1	エ 終点[I] ← 終点P+1

eに関する解答群

ア I+1, L ＜ N, 1	イ I+1, L ≦ N, 1
ウ N, L ≧ I+1, −1	エ N, L ＞ I+1, −1

設問3　次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。

このアルゴリズムでは，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$_1$に値 -∞ を，終点$_N$に値 +∞ をそれぞれ設定している。このような設定をすることの利点の一つに，［　f　］という特徴が挙げられる。

また，このアルゴリズムでは，空きリスト中の組の個数が変化する。領域中のセル数が E 個であるとする。このとき，空きリスト中の組の個数は，最大で［　g　］となる。また，E 個の全てのセルが“割当済み”となったとき，空きリスト中の組の個数は，［　h　］となる。ここで，整数同士の除算では，商の小数点以下を切り捨てる。

fに関する解答群

ア　空きリストが空（組の個数が 0）にならない

イ　関数 Free の実行時に空きリスト中の組の個数が 2 以上であることが保証される

ウ　始点$_1$又は終点$_N$の値が変わらない限り領域中に“空き”セルが残っている

エ　領域中の一つの連続した“空き”セルが幾ら長くても一つの組で表せる

g，hに関する解答群

ア　1	イ　2	ウ　E÷2＋1
エ　(E＋1)÷2＋1	オ　E＋1	カ　E＋2