平成20年度　春期
ソフトウェア開発技術者
午後Ⅱ　問題

試験時間	15:30 ～ 16:30（1時間）

注意事項

1. 試験開始及び終了は，監督員の時計が基準です。監督員の指示に従ってください。
2. 試験開始の合図があるまで，問題冊子を開いて中を見てはいけません。
3. この注意事項は，問題冊子の**裏表紙**に続きます。必ず読んでください。
4. **答案用紙への受験番号などの記入は，試験開始の合図があってから始めてください。**
5. 問題は，次の表に従って解答してください。

問題番号	問1
選択方法	必須

6. 答案用紙の記入に当たっては，次の指示に従ってください。
   (1) B又はHBの黒鉛筆又はシャープペンシルを使用してください。
   (2) **受験番号欄**に，**受験番号**を記入してください。正しく記入されていない場合は，採点されません。
   (3) **生年月日欄**に，受験票に印字されているとおりの**生年月日**を記入してください。正しく記入されていない場合は，採点されないことがあります。
   (4) 解答は，問題番号ごとに指定された枠内に記入してください。
   (5) 解答は，丁寧な字ではっきりと書いてください。読みにくい場合は，減点の対象になります。

注意事項は問題冊子の裏表紙に続きます。
こちら側から裏返して，必ず読んでください。

問1　パズルの解法に関する次の記述を読んで，設問1～5に答えよ。

あるルールに従って，盤面上の空欄に数字を入れるパズルを解くアルゴリズムについて考える。このパズルは図1に示すように，縦横それぞれ9マスからなる盤面で，幾つかのマスには数字が入れられている。この状態から，数字の入っていない各マスに，1～9のうちのどれか一つの数字を入れていく。このとき，横1列，縦1列，及び太線で囲まれた3×3の小枠のどのブロックにおいても，1～9の数字が一つずつ入ることが，このパズルのルールである。図1にパズルの問題例を，図2に図1の解を示す。

		3	9			7	6	
	4				6			9
6				1				4
2			6	7			9	
		4	3		5	6		
	1			4	9			7
7				9		2		1
3			2				4	
	2	9			8	5		

図1　問題例

1	5	3	9	8	4	7	6	2
8	4	2	7	3	6	1	5	9
6	9	7	5	1	2	8	3	4
2	3	8	6	7	1	4	9	5
9	7	4	3	2	5	6	1	8
5	1	6	8	4	9	3	2	7
7	6	5	4	9	3	2	8	1
3	8	1	2	5	7	9	4	6
4	2	9	1	6	8	5	7	3

図2　図1の解

このパズルを解くアルゴリズムを，ステップ1～3の順で考える。

〔ステップ1〕

(1)　盤面全体で，まだ数字が確定していないすべてのマスに対して(2)～(6)の手順を実施する。

(2)　数字が確定していないマスに入れる数字の候補を，1～9のすべての数とする。

(3)　そのマスが置かれている横1列のブロックのマスを見て，既にほかのマスに入っている数字を，そのマスの候補から除く。

(4)　そのマスが置かれている縦1列のブロックのマスを見て，既にほかのマスに入っている数字を，そのマスの候補から除く。

(5)　そのマスが置かれている3×3の小枠のブロックのマスを見て，既にほかのマスに入っている数字を，そのマスの候補から除く。

(6) (3)～(5)によって，候補の数字が一つになったら，そのマスにはその数字を入れる。

(7) 新たに数字を入れたマスが存在する間，(1)～(6)の手順を繰り返す。

〔ステップ2〕

横1列，縦1列，3×3の小枠の各ブロックで，数字が確定していないすべてのマスにおいて，そのマスに入れる数字の候補を考える。このとき，各ブロック内において，ある数字がただ一つのマスの候補にしか含まれていない場合には，その数字をそのマスに入れる。

例えば，あるブロックで数字が確定していないマスが三つあり，それらのマスをそれぞれA，B，Cとする。このとき，マスAの候補が2，3，6，マスB，Cの候補がともに2，3とすると，6はAのマスにしか入る可能性はないので，Aのマスの数字として6を入れる。

すべてのブロックに対する操作の中で，新たに数字を入れたマスがあれば，ステップ1に戻る。

〔ステップ3〕

ステップ1，2でも数字が確定しないマスがある場合，数字が確定していないマスを総当たりで探索する。ただし，ここではこのアルゴリズムについては考えない。

このパズルを解くために，表1に示す定数と配列，及び図3のデータ構造を使用する。配列の添字は1から始めるものとする。

表1　定数と配列

定数，配列名	内容
ncol	盤面の横列，縦列の数。ここでは9。
sqncol	小枠の横列，縦列の数。ここでは3。
item[ ]	各マスに入っている数字や，入れる数字の候補の情報をもつ配列。図3に構造を示す。変数 number はマスに入っている数字を示す。number の値は，マスに入れる数字が確定している場合にはその数字，確定していない場合は 0 となる。配列 candidate[ ]はマスに入れる候補となる数字を示す。candidate[k]の値は，数字 k が候補のときは TRUE，k が候補でないときは FALSE となる。変数 number，candidate[ ]は，それぞれ item[ ].number，item[ ].candidate[ ]と表記する。

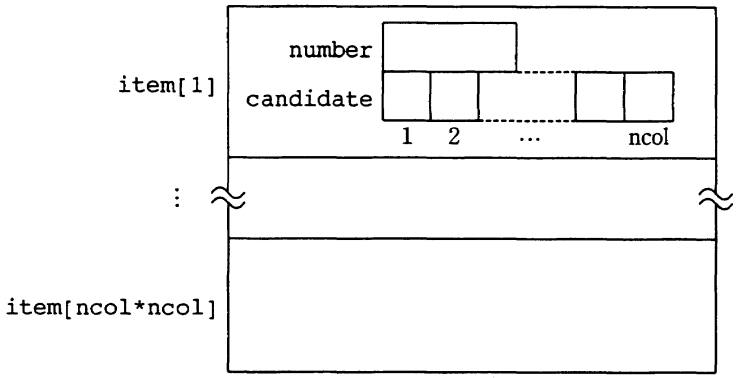


図3　データ構造

次に，このパズルを解くために使用する関数のインタフェース，機能，戻り値を表2に示す。ただし，横列（行）は上から順に1行，2行，…，ncol行，縦列（列）は左から順に1列，2列，…，ncol列と指定する。このとき，item[ ]は，1行1列，1行2列，…，ncol行(ncol−1)列，ncol行ncol列の順に，各マスの情報をもつ。

表2　使用する関数

関数のインタフェース	機能又は戻り値
pairToIndex(i, j)	i行j列のマスの情報をもつ配列itemの要素の添字。
getNumber(i, j)	i行j列のマスに入っている数字。未確定の場合は0。
isCandidate(i, j, k)	i行j列のマスでkが候補ならTRUE，候補でないならFALSE。
setNumber(i, j, n)	i行j列のマスに数字nを入れる。
countCandidate(i, j)	i行j列のマスにおいて，候補となる数字の個数。
extractCandidate(i, j)	i行j列のマスの候補の数字が一つのとき，その候補の数字。それ以外のときは0。
removeCandidate(i, j)	i行j列のマスで，すべての数字を候補にした後で，同じ縦列，横列，小枠に入っている数字を候補から外す。
evaluateCandidate( )	全マスについてremoveCandidateを行う。

表2に示した関数のうちの一部のプログラムを図4に示す。また，〔ステップ1〕によって解を得る関数`first_solve`を図5に示す。

```
function pairToIndex(i, j)
  return [  ア  ]
endfunction

function countCandidate(i, j)
  m ← 0
  for (n を 1 から [  イ  ] まで 1 ずつ増やす)
    if (item[pairToIndex(i, j)]. [  ウ  ] が TRUE に等しい)
      m に 1 を加える
    endif
  endfor
  return m
endfunction

function setNumber(i, j, n)
  if (item[pairToIndex(i, j)].number が 0 より大きい)
    return -1
  endif
  [        エ        ] ← n
  return 0
endfunction
```

図 4　一部の関数のプログラム

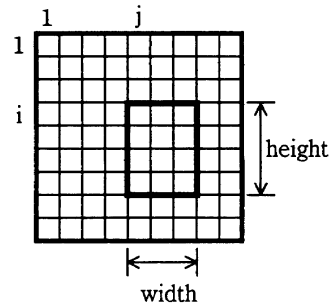
```
function first_solve()
  count ← 100                                    // count に正数をセット
  while ( [  オ  ] が 0 より大きい)
    count ← 0
    for (i を 1 から ncol まで 1 ずつ増やす)
      for (j を 1 から ncol まで 1 ずつ増やす)
        if (getNumber(i, j)が [  カ  ] )          // 数字が未確定ならば
          removeCandidate(i, j)
          k ← [  キ  ] (i, j)
          if (k が [  ク  ] )                     // 候補が一つならば
            setNumber(i, j, k)                   // その候補の数字を設定
            count に 1 を加える
          endif
        endif
      endfor
    endfor
  endwhile
endfunction
```

図 5　ステップ 1 のプログラム

次に，〔ステップ 2〕のアルゴリズムを実行する関数 second_solve について考える。ただし，〔ステップ 2〕中の“新たに数字を入れたマスがあれば，ステップ 1 に戻る”部分のプログラムはここでは考えないものとする。

second_solve 内で使用する関数 blockCheck のインタフェースと機能，blockCheck で使用する配列 ar の内容を，表 3 に示す。また，second_solve と blockCheck のプログラムをそれぞれ図 6，7 に示す。

表 3　ステップ 2 で使用する関数と配列

インタフェース又は配列名	機能，内容
blockCheck (i, j, height, width)	i 行 j 列のマスを左上の角とする高さ height，幅 width 列分のブロック内で，数字が未確定のすべてのマスで候補となる数字を考え，ある数字がブロック内の一つのマスでしか候補になっていない場合，その数字をそのマスに入れる。 
ar[k]	数字 k が候補になっているマスの個数。

```
function second_solve()
  for (i を 1 から ncol まで 1 ずつ増やす)          // 横 1 列のブロックをスキャン
    blockCheck( [ケ] )
  endfor
  for (j を 1 から ncol まで 1 ずつ増やす)          // 縦 1 列のブロックをスキャン
    blockCheck( [コ] )
  endfor
  for (i を 1 から(ncol-sqncol+1)まで sqncol ずつ増やす)   // 小枠をスキャン
    for (j を 1 から(ncol-sqncol+1)まで sqncol ずつ増やす)
      blockCheck( [サ] )
    endfor
  endfor
endfunction
```

図 6　ステップ 2 のプログラム

```
function blockCheck(i, j, height, width)
  evaluateCandidate( )                    // 盤面全体で候補を再評価
  for (k を 1 から ncol まで 1 ずつ増やす)
    ar[k] ← 0                             // 配列の初期化
  endfor

  for (s を i から [  シ  ] まで 1 ずつ増やす)
    for (t を j から [  ス  ] まで 1 ずつ増やす)
      if ( [        セ        ] )         // 数字が未確定ならば
        for (k を 1 から ncol まで 1 ずつ増やす)
          if (isCandidate(s, t, k)が [  ソ  ] )
            ar[k]に 1 を加える            // 数字 k が候補のマスの数を増やす
          endif
        endfor
      endif
    endfor
  endfor

  for (k を 1 から ncol まで 1 ずつ増やす)
    if (ar[k] が 1 に等しい)              // k を候補とするマスが一つの場合
      // ブロックのどこに候補の数字があるかを探す
      for (s を i から [  シ  ] まで 1 ずつ増やす)
        for (t を j から [  ス  ] まで 1 ずつ増やす)
          if ( [        セ        ] かつ [        タ        ] )
            [        チ        ]        // 該当するマスに数字を入れる
            goto out                      // ラベル out へジャンプ
          endif
        endfor
      endfor
    endif
out:                                      // ラベル
  endfor
endfunction
```

図 7　blockCheck のプログラム

設問1　本文中で説明されたパズルのルールに従って，次の盤面中の(a)～(j)のマスに入れる数字を答えよ。

1	(a)	7	8	(b)	2	9	(c)	5
(d)	5	(e)	9	7	(f)	1	(g)	3
8	9	(h)	1	(i)	5	(j)	2	7
2	1	5	3	6	9	7	4	8
9	6	4	7	5	8	3	1	2
3	7	8	4	2	1	5	9	6
7	8	3	2	1	4	6	5	9
6	2	1	5	9	3	8	7	4
5	4	9	6	8	7	2	3	1

設問2　図4中の［　ア　］～［　エ　］に入れる適切な字句を答えよ。

設問3　図5中の［　オ　］～［　ク　］に入れる適切な字句を答えよ。

設問4　図6及び図7中の［　ケ　］～［　チ　］に入れる適切な字句を答えよ。

設問5　図4に示すプログラム `setNumber` 内の下線部では，この関数が呼び出されることを想定していない場合であることを示すために，`-1` を返している。どのような場合か，30字以内で述べよ。

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

7. 途中で退室する場合には，手を挙げて監督員に合図し，答案用紙が回収されてから静かに退室してください。

退室可能時間	16:10 ～ 16:20

8. **問題に関する質問にはお答えできません。**文意どおり解釈してください。
9. 問題冊子の余白などは，適宜利用して構いません。
10. 試験中，机上に置けるもの及び使用できるものは，次のものに限ります。

    なお，会場での貸出しは行っていません。

    受験票，黒鉛筆又はシャープペンシル，鉛筆削り，消しゴム，定規，時計（アラームなど時計以外の機能が付いているものは不可），ハンカチ，ティッシュ

    これら以外は机上に置けません。使用もできません。
11. 試験終了後，この問題冊子は持ち帰ることができます。
12. 答案用紙は，いかなる場合でも，すべて提出してください。回収時に提出しない場合は，採点されません。
13. 試験時間中にトイレへ行きたくなったり，気分が悪くなったりした場合は，手を挙げて監督員に合図してください。