問 1

ゲームプログラムの作成に関する次の記述を読んで，設問 1 ～ 4 に答えよ。

人(以下，プレイヤという)とコンピュータが対戦する数当てゲームのプログラムを作ることになった。ゲームのルールは，次のとおりである。

(1)対戦者(プレイヤとプログラム)は，四つの異なる数字からなる 4 けたの数をそれぞれ決める。ここでは，これを正解と呼ぶことにする。正解の先頭(最上位けた)は 0 でも構わない。

(2)相手の正解を交互に推測して，先に当てた方が勝ちとする。同じ推測回数で当てたときは引き分けとする。

(3)互いに相手が推測した数を受け取ると，その数と自分の正解を比べ，数字もけたも一致している数字の個数(hit)と，数字は含まれているがけたが違う数字の個数(miss)を返す。例えば，正解 1357 に対し 5310 を推測値として受け取ると，hit は 1，miss は 2 を返す。

数当てゲームのプログラムの概略は，次のとおりである。

```
1　乱数を使って正解を決める。
2　どちらかが相手の正解を当てるまで，次を繰り返す。
  2-1　正解との比較処理
        プレイヤの推測値を受け取り，正解と比較し，hit と miss を返す。
  2-2　推測処理
      2-2-1　それまでに受け取った hit と miss を基に次の推測値を決め，
             プレイヤに渡す。
      2-2-2　プレイヤの計算した hit と miss を受け取る。
```

以下では，関数の中で定義している変数は局所変数とし，関数の引数は，断りのない限り，値渡しとする。配列の添字は 0 から始まるものとし，配列の要素は，0 番目，1 番目と数える。

正解を決めるための関数を図 1 に示す。この関数では 0 ～ 9 の数をランダムに一つずつ生成し，10 個の要素からなる配列 tab を用いて数字に重複のない 4 けたの数を求めている。random は自然数をランダムに生成する関数，% は剰余演算子であり，配列 tab を次のように利用している。

(1)すべての要素の初期値は 0 とする。

(2)ランダムに生成した 1 けたの数 a に対して，tab[a] の値が初期値 0 のままなら，a は初めて現れた数なので，tab[a] に 1 を入れる。

(3)tab[a] の値が 1 なら，その値は変更しない。

```
function set_answer()
    int tab[10], n, i, a
    tab の要素をすべて 0 にする
    n = 0
    for(i を 1 から 4 まで)
        repeat
```

```
            a = random() % 10
        until(tab[a]が 0 に等しい)
        tab[a] = 1
        n = 【      a      】
    endfor
    return  n
endfunction
```

図 1　関数 Set_answer

次に，hit と miss を計算するアルゴリズムを考える。

まず，二つの 4 けたの数 x と y の類似度 similarity を，二つの数の hit と miss を用いて，

similarity(x, y) = hit * 10 + miss

と定義する。similarity の値と(hit，miss)の値の組は 1 対 1 に対応するので，プログラムの中では hit と miss の代わりに similarity を使う。

similarity の計算は，プレイヤの推測値を正解と比較する場合と，プログラムが推測値を決める場合に利用する。この計算は頻繁に行うので，効率の良い実行方式を考える。

実行効率を上げるため，ゲーム中に行う similarity の計算の前に，比較対象になる一方の数を配列表現と呼ぶ形式に変換しておく。この変換処理では，対象とする数を d4d3d2d1 とすると，配列の di 番目の要素には i を，そのほかの要素には 0 を格納する。数 y を配列表現に変換する関数を図 2 に示す。pos は 10 個の要素からなる配列で，参照渡しである。

```
function make_pos(y, pos)
    int i,m
    pos の要素をすべて 0 にする
    for(i を 1 から 4 まで)
        m = y % 10
        y = y / 10
        pos[【      b      】] = i
    endfor
endfunction
```

図 2　関数 make_pos

数 y の配列表現(pos)を利用すると，similarity(x，y)は，図 3 の関数 sim で求めることができる。

```
function sim(x, pos)
    int hit, miss, i, p
```

```
    hit = 0
    miss = 0
    for(i を 1 から 4 まで)
        p = x % 10
        x = x / 10
        if(【     c     】)
            if(【     d     】)hit = hit +1
            else   miss = miss + 1
            endif
        endif
    endfor
    return hit * 10 + miss
endfunction
```

図 3　関数 sim

次に推測のアルゴリズムを考える。

四つの異なる数字からなる 4 けたの数をすべて生成すれば，その中には正解が必ず含まれるので，それを前もって生成して配列 nums に入れておくことにする。

プレイヤの正解を当てるために推測値を選ぶアルゴリズムは次のとおりとする。nums から順次，数を取り出し，それまでに得た similarity の情報に矛盾しない数を次の推測値とする。

例えば，3 回目の推測値を選ぶ場合を考える。正解とそれまでの 2 回の推測値との similarity の値が次のとおりだったとする。

	推測値	similarity
１回目	1234	12
２回目	1345	02

このとき，nums から取り出された次の数が 2304 だったとする。2304 が次の推測値として適切かどうか計算してみると，similarity(2304, 1234) = 12, similarity(2304, 1345) = 11 となり，2 回目の結果と矛盾するので，2304 は次の推測値としては適切でない。

そこで，更に nums から数を取り出し，その値が 0432 だったとする。0432 が次の推測値として適切かどうか計算してみると，similarity(0432, 1234) = 12, similarity(0432, 1345) = 02 で今までの結果と矛盾しない。したがって，0432 を次の推測値とする。

これらの考察をまとめると，k 回目の推測値を求める関数は図 4 のようになる。ここで，配列 nums は正解の候補の入った配列，guess[i-1] は i 回目の推測値の配列表現，log[i-1] は i 回目の推測値と正解の similarity の値である。index は，nums の要素の中で，これより前には正解が存在しない位置を指している添字で，初

期値は -1 である。

```
function   make_quess(k)
    int j
    index = index + 1
    for()                                          //無限ループ
        j = 0
        while (j<k-1 かつ sim(nums[index], guess[j])が log[j] に等しい)
            【     e     】
        endwhile
        if(j<k-1)
            【     f     】
        else
            return nums[index]
        endif
    endfor
endfunction
```

図 4　関数 make_guess

先に述べた各関数を使うと，数当てゲームのプログラムは図 5 のようになる。nums，guess，log，index は大域変数である。

```
int log[M], guess[M][10], nums[N], index      //M，N は配列サイズ
main()
    char   r
    nums に，四つの異なる数字からなる 4 けたのすべての数を入れる
    for()                                          //ゲームを繰り返す
        p1ay()
        write "ゲームを続けますか?   Y/N"
        read   r
        if(r が "N"に等しい)                        //ゲーム終了をプレイヤが選択
            exit
        endif
    endfor
endmain
function   play()
    int   ans, pos_a[10], i, r, g, hit, miss, guess_person
    ans = set_answer()
    make_pos(ans, pos_a)
    index = -1
```

```
    i = 1
    for()                                          //当たるまで繰り返す
        read   guess_person                        //プレイヤの推測値を読む
        r = sim(guess_person, pos_a)               //similarity を計算
        write   r / 10, r % 10                     //結果を表示
        g = make_guess(i)                          // i 回目の推測値を計算
        Write   g                                  //推測値を表示
        read hit, miss                             //推測の結果を読む
        if(hit が 4 に等しい又は r が 40 に等しい)   //当たったか
            break
        endif
        log[i-1] =10 * hit + miss                  //結果を記録
        make_pos(g, guess[i-1])                    //推測値を記録
        i = i+ 1
    endfor
    if(【     g     】)                             //プレイヤの勝ち
        write "あなたの勝ちです"
    elseif(【     h     】)
        write "私の勝ちです"                       //コンピュータの勝ち
    else
        write "引き分けです"                       //引き分け
    endif
endfunction
```

図 5　数当てゲームのプログラム

設問 1　図 5 のプログラムで，配列 nums の大きさ N は最低幾つでなければならないか。

設問 2　図 1 ～ 5 中の【　　a　　】～【　　h　　】に入れる適切な字句を答えよ。

設問 3　プレイヤがしばらく対戦を進めたところ，プログラムから応答がなくなった。調査したところ，プレイヤが hit 又は miss の値を間違って返したので，正解になる候補が見付からないことが原因であった。この場合には，make_guess が -1 を返すようにしたい。

図 4 の関数 make_guess にどのような判定を追加すればよいか。20 字以内で述べよ。

設問 4　図 5 のプログラムのアルゴリズムでは，プレイヤがコンピュータとの対戦を繰り返しても，コンピュータは毎回同じ推測を繰り返すので，プレイヤはコンピュータが当てにくい数を見付けられるようになる。

これを回避するために，プログラムで毎回，異なる推測の系列を使いたい。毎回異なる推測系列を使うには，main から play を呼ぶ直前で配列 nums の値にある操作を行えばよい。操作内容を 20 字以内で述べよ。

【解答例】

設問 1 5040

設問 2 a-n*10+a b-m c-0 e-j=j+1 f-index=index+1 g-hit<4 h-r<40

設問 3 index **が** N **を超えたかどうかの判定**

設問 4 **配列** nums **内の各値の格納順序を変える**