情報科学 共通問題 (2012 年度冬学期試験)

[科目名: 情報科学, 試験実施日: 2013 年 2 月 13 日第 2 限, 答案用紙: 1 部, 計算用紙: 1 枚, 持込み: 一切不可]

- 内容に関する質問は受け付けない。問題の記述があいまいな場合は、適切な仮定を置いて解答し、どのような仮定を置いたかを明記せよ。

以下の問題で用いられている Ruby の式の意味は次のとおりである。x `&&` y は x と y の値がともに真の場合にのみ真となる。x `||` y は x と y の値がどちらか真の場合にのみ真となる。`!`x は x の真偽を逆転させる。また、関数 `min`(x`,`y), `max`(x`,`y) は x, y のそれぞれ小さい方、大きい方の値を返す。関数 `abs`(x) は x の絶対値を求める。関数 `make1d`(n) は大きさ n の配列を作る。問題においても解答の際も、これらの関数は定義済みであるとしてよい。

問題 1

(a) 右の関数定義の下で `f(5)` が何を計算するか述べよ。

(b) 上の小問の関数 `f` は再帰呼び出しを使って書かれている。再帰呼び出しを使わず `while` や `if`, `for` だけを使って同じ計算をするように `f` を書き直せ。

```
def f(n)
    if n >= 2
        n * f(n - 1)
    else
        1
    end
end
```

(c) 右の関数定義の下で `g(5)` が何を計算するか述べよ。

```
def g(n)
    if n >= 2
        g(n - 1) + g(n - 2)
    else
        1
    end
end
```

(d) 上の小問の関数 `g` を、再帰呼び出しを使わないように書き直したものが右の関数 `h` である。空欄を埋めよ。

```
def h(n)
    result = make1d(n+1)
    i = [ア]
    while [イ]
        if i >= 2
            result[i] = [ウ]
        else
            result[i] = 1
        end
        i = [エ]
    end
    result[ [オ] ]
end
```

問題 2　配列 a に、n 個の整数値が入っているとする ($n \geq 2$)。a の中の任意の 2 つの値の差の絶対値の最小値を求める問題について考える。例えば a = [9, 3, 5] の場合には 2 と答えるようなものである。この問題を解くアルゴリズムについて以下の小問に答えよ。

(a) 計算量が $O(n^2)$ であるようなアルゴリズムを、文章あるいは Ruby の関数として書け。

(b) 配列 a が大きさ順に整列されていたと仮定する。この仮定の下で計算量で $O(n)$ となるアルゴリズムを考え、文章あるいは Ruby の関数として書け。

(c) 配列 a が整列されているとは限らないときのアルゴリズムのうち、(a) よりも計算量のオーダーが良いものを 1 つ考えよ。その計算量のオーダーと理由を書け。

問題 3

(a) 以下に示すのは Gauss-Jordan 法によって n 変数の連立 1 次方程式を解く関数 gj(a) である。[ア] と [イ] に適切な式を入れて、プログラムを完成させよ。

```
def gj(a) # Gauss-Jordan method without pivoting
  row = a.length()
  col = a[0].length()
  for k in 0..(col-2)
    akk = a[k][k]
    for i in 0..(col-1)
      a[k][i]=a[k][i] * [ア]
    end
    for i in 0..(row-1)
      if i != k
        aik = a[i][k]
        for j in k..(col-1)
          a[i][j] = a[i][j] - [イ]
        end
      end
    end
  end
  a
end
```

(b) (a) のプログラムによる計算の際に生じる可能性がある誤差について説明せよ。

(c) Pivoting について説明し、(a) のプログラムをどのように修正すれば pivoting ができるかを示せ。配列 a で k 列目の絶対値が最大となる行を k 行目以降から探す関数 maxrow(a,k) と、配列 a の k 番目と max 番目を入れ替える swap(a,k,max) は定義済みとして使ってよい。

(d) 不能 (解が存在しない場合) や不定 (複数の解が存在する場合) となる連立一次方程式に、(c) で修正したプログラムを適用した時に、不能や不定であることができるだけ早く判定でき、計算を打ち切れるようにプログラムを修正せよ。「計算を打ち切る」には関数 abort() を呼ぶだけでよい。関数 abort() を呼ぶとプログラムの実行はそこで止まる。

問題 4　文字列の配列が二つ与えられているとする。これらを a と b とする。a と b の大きさ（文字列の数）は同じであるとする。たとえば、a = ["0", "11"]、b = ["101", "1"] とする。

a から重複を許して文字列をいくつか選んで連結した結果と、b から対応する添字の文字列を選んで同じ順序で連結した結果が、全く同じ文字列になるようにしたい。たとえば、上の例の場合、a の $1, 0, 1$ 番目つまり a[1] と a[0] と a[1] を連結すると "11011" が得られる。これに対して、b の $1, 0, 1$ 番目つまり b[1] と b[0] と b[1] を連結すると、全く同じ文字列である "11011" が得られる。

(a) a = ["1", "0", "00"]、b = ["0", "011","0"] としたとき、連結して全く同じ文字列が得られるような文字列の選び方を添字の並びとして答えよ。

(b) 次のプログラムは、上の問題を解こうとするものである。関数 post の最初の二つの引数として二つの配列を指定する。三番目の引数としては配列の大きさを指定する。

```
def post(a, b, m, n, as, bs)
  if as != "" && as == bs
    as
  else
    if n == 0
      ""
    else
      k = 0
      p = ""
      while p == "" && k < m
        p = post(a, b, m, n-1, as+a[k], bs+b[k])
        k = k + 1
      end
      p
    end
  end
end
```

なお、演算子 + は、文字列に対しては、連結の操作を意味する。たとえば、"010" + "00" は "01000" になる。"" は空文字列を表し、"" + "00" は "00" になる。また、比較演算子 ==（!=）は、文字列に対しては、同じ文字列であるか（ないか）を判定する。

post(["0", "11"], ["101", "1"], 2, 1, "110", "1101") を実行すると何が返るか。

(c) 小問 (a) の例に対して、上の問題を解くためには、関数 post をどのように呼び出せばよいか。また、結果として何が返るか。

(d) どのように文字列を選んでも全く同じ文字列を作れない場合に対して、関数 post の m と n に関する計算量のオーダーを答えよ。ただし、文字列の演算は文字列の長さによらずに一定時間しかかからないものと仮定する。

以上