情報科学　共通問題（2014 年度冬学期試験）

[試験日時：2015 年 2 月 9 日第 4 限，答案用紙：1 部，計算用紙：1 枚，持ち込み一切不可]

内容に関する質問は受け付けない。問題の記述が曖昧な場合は、適切な仮定をおいて回答し、どのような仮定をおいたのか明記せよ。

以下の問題で用いられている make2d(m,n) は $m \times n$ の二次元配列を作る。これは定義済みとしてよい。

問題 1

今、ベクトル $(a_0, \ldots, a_{n-1})$ を配列 $[a_0, \ldots, a_{n-1}]$ で表すことにする。

1. 図 1 中の空欄 (A)〜(D) を埋め、ベクトルの内積を計算する関数 inner とベクトルの和を計算する関数 add を完成させよ。これらは再帰関数である。引数 a、b は同じ長さの配列とする。ただし同じ記号の欄は同じ内容である。

   変数 a が配列を表すとき a.length は配列の長さを表す。また、文字列同様の表記で部分配列を得られる。例えば

   ```
   p = [11, 13, 17, 19, 23]
   q = p[2..4]
   ```

   を計算すると q は配列 p の 2 番目から 4 番目までの要素からなる配列である（配列の先頭は 0 番目であることに注意）。つまり q は [17, 19, 23] である。

   また、「配列+配列」は配列の連結を意味する。例えば [1]+[2] は [1, 2]、[1, 2] + [3, 4, 5] は [1, 2, 3, 4, 5] である。[ ] は要素数 0 の空の配列である。[ ] + [1] は [1]、[ ] + [1, 2] は [1, 2] である。

2. $n \times n$ 行列を 2 次元配列 $[[a_{00}, \ldots, a_{0n'}], [a_{10}, \ldots, a_{1n'}], \ldots, [a_{n'0}, \ldots, a_{n'n'}]]$ （$n' = n-1$）で表すことにする。ベクトル v と行列 $A = [[a_{00}, \ldots, a_{0n'}], \ldots, [a_{n'0}, \ldots, a_{n'n'}]]$ の積はベクトルであり、i 番目の要素の値が v とベクトル $[a_{0i}, \ldots, a_{n'i}]$ の内積である。

   図 1 中の空欄 (E)〜(G) を埋めて長さ n のベクトル v と $n \times n$ の行列 a の積を計算する関数 vmmult(v,a) を完成させよ。空欄を埋めるにあたって上で定義した関数 add や inner を用いてもよい。ここで a.transpose は変数 a が表す行列の転置行列（縦と横が入れ替わった行列）を意味する。

3. 行列 $A = [[a_{00}, \ldots, a_{0n'}], \ldots, [a_{n'0}, \ldots, a_{n'n'}]]$ と $B = [[b_{00}, \ldots, b_{0n'}], \ldots, [b_{n'0}, \ldots, b_{n'n'}]]$ の積は行列であり、i 行 j 列の要素の値がベクトル

```
def inner(a, b)
  if (A)
    0
  else
    a[0]*b[0] + inner(a[(B)], b[(C)])
  end
end

def add(a, b)
  if (A)
    [ ]
  else
    [(D)] + add(a[(B)], b[(C)])
  end
end

def vmmult(v, a)
  vmmult1(v, a.transpose)
end

def vmmult1(v, a)
  if (E)
    [ ]
  else
    [(F)] + vmmult1(v,a[(G)])
  end
end

def mmmult(a, b)
  mmmult1(a, b.transpose)
end

def mmmult1(a, b)
  if (H)
    [ ]
  else
    [(I)] + mmmult1(a[(J)],b)
  end
end
```

図 1: 関数 inner・add・vmmult・mmmult

[a_{i0},...,$a_{in'}$] とベクトル [b_{0j},...,$b_{n'j}$] の内積である。

図 1 中の空欄 (H)〜(J) を埋めて $n \times n$ の行列同士の積を計算する関数 mmmult を完成させよ。空欄を埋めるにあたって上で定義した関数 add や inner、vmmult を使ってもよい。

問題 2

A さんは現在金貨を 5 枚を持っており、これからあるゲームを繰り返し行う。1 回のゲームで勝てば金貨を 1 枚もらえ負ければ 1 枚失うものとし、A さんが勝つ確率を p、負ける確率を q とする ($p+q=1$)。A さんの金貨が 0 枚または 10 枚になった時点で繰り返しを終了する。

第 t 回目のゲーム終了時に A さんが金貨を a 枚持っている確率を $P(t,a)$ と書く（開始時は $t=0$ とする)。ただし、$P(t,0)$ や $P(t,10)$ は「ちょうど t 回目に 0 枚（または 10 枚）になった確率」を表し、i 回目（$i<t$）に 0 枚（または 10 枚）になり以降のゲームを行わなかった場合は含まれないとする。

以下の問に答えよ。

1. $P(t,a)$ を第 $t-1$ 回目終了時の確率を用いて表せ。
2. 問 1 の関係式を利用し、再帰的に $P(t,a)$ を計算するプログラムを作成したところ、t が小さな場合でも計算終了までにかなり時間がかかった。その理由を考察し数行で説明せよ。
3. 次に、配列を用いて $P(t,a)$ を計算する関数 P_loop(t,a)（図 2）を作成した**（このプログラムは出題時のものからより適切なものへと修正されている）**。この関数は prob[i][j] $= P(i,j)$ となるよう 2 次元配列 prob を埋める。(A)〜(E) を埋めよ。ただし同じ記号の欄は同じ内容である。また p および q は変数 p および q として参照できるとする。
4. 問 3 で作ったプログラム P_loop(t,a) の計算量を評価したい。一般に A さんの金貨の枚数の上限を n 枚としたとき（上では $n=10$)、P_loop(t,a) の計算量のオーダーを n と t を用いて表せ。導出の過程も簡単に説明すること。
5. 計算量のオーダーとは何か、オーダーを考えることがどのように役に立つかを 2 行程度で簡単に説明せよ。

```
def P_loop(t,a)
  prob = make2d(t+1,11)
  for j in 0..10
     if j == 5
       prob[ (A) ][j] = 1
     else
       prob[ (A) ][j] = 0
     end
  end
  for i in 1..t
    for j in 0..10
      if j == 0 || j == 1
        prob[i][j] = (B)
      else if j == 9 || j == 10
             prob[i][j] = (C)
           else
             prob[i][j] = (D)
           end
      end
    end
  end
  (E)
end
```

図 2: 関数 P_loop(t,a)

問題 3

次の小問 1〜8 に答えよ。

ECCS 環境の irb で簡単な計算を行った結果を図 3 に示す。これについて以下の問 1〜6 に答えよ。

1. 図 3 の下線部 (a) が理論上の値である 1.001 と異なる理由を 1 行程度で述べよ。
2. 図 3 の下線部 (b) が理論上の値である 1 と異なる理由を 1 行程度で述べよ。
3. 図 3 の下線部 (b) と下線部 (c) が異なる理由を 1 行程度で述べよ。
4. 図 3 の下線部 (c) と下線部 (d) が大きく異なる理由を 1 行程度で述べよ。

```
irb(main):001:0> a = 1000
=> 1000
irb(main):002:0> 1/a + 1
=> 1(a)
irb(main):003:0> a = 1000.0
=> 1000.0
irb(main):004:0> 1/a + 1
=> 1.001
irb(main):005:0> (1/a + 1 - 1) * a
=> 0.9999999999998899(b)
irb(main):006:0> (1/a**5 + 1 - 1) * a**5
=> 1.1102230246251565(c)
irb(main):007:0> (1/a**6 + 1 - 1) * a**6
=> 0.0(d)
irb(main):008:0> a = 1024.0
=> 1024.0
irb(main):009:0> (1/a**4 + 1 - 1) * a**4
=> (A)
irb(main):010:0> (1/a**6 + 1 - 1) * a**6
=> (B)
```

図 3: irb での計算結果

```
def calc(n)
  s = 0
  for i in 1..n
    x = rand()
    y = rand()
    if sqrt(x**2 + y**2) > 0.5
      s = s + 1
    end
    if sqrt(x**2 + (1-y)**2) > 0.5
      s = s + 1
    end
    if sqrt((1-x)**2 + y**2) > 0.5
      s = s + 1
    end
    if sqrt((1-x)**2 + (1-y)**2) > 0.5
      s = s + 1
    end
  end
  s * 1.0 / n
end
```

図 4: 関数 f

5. 空欄 (A) を埋めよ。またそのような結果となる理由を 1 行程度で説明せよ。
6. 空欄 (B) を埋めよ。またそのような結果となる理由を 1 行程度で説明せよ。

次に、図 4 に示す関数 calc について以下の問 7・8 に答えよ。ただし sqrt(v) は v の平方根を求める関数であり、rand() は 0 以上 1 未満の実数を一様分布に従い発生させる関数である。ただし、rand 関数は理想的な乱数生成器であり、また数値誤差はないものとしてよい。

7. calc(1) を実行し、2 回の rand 関数でそれぞれ 0.4 と 0.2 が得られたとき、この calc(1) の実行結果を答えよ。
8. n を大きくしてゆくに従い、calc(n) の結果は平均的にはある値に収束する。その値を円周率を π として答えよ。

問題 4

N 人の盗賊が密談をしており、これから一人ずつ順に出てきてそれぞれバラバラの隠れ家に戻る。これに対し、M ($0 < M \leq N$) 人の捜査官が、盗賊を隠れ家まで尾行し取り押さえるべく 1 列に待機している。司令官であるあなたはどの盗賊にどの捜査官をつけるかを決め、できる限り多額の盗品を回収したい。i 番目の盗賊から j 番目の捜査官が回収できる金額は二次元配列 C[i][j] で知ることができる。最初の盗賊および捜査官は 0 番目であることに注意せよ。また、回収できる金額は正の整数である。尾行の都合で 1 人の盗賊につけられる捜査官はたかだか 1 人である。さらに、捜査官は待機場所の都合で列の先頭から順に出発する必要があり、士気を保つために全員が誰かの担当になる必要がある。

以上をふまえ以下の問に答えよ。なお、プログラム中では 2 次元配列 C は常に利用可能だとしてよい。

例: N=5, M=2

可能な配置の例

盗賊　01234
捜査官 --0-1
回収額 C[2][0]+C[4][1]

不可能な配置の例 (1)

盗賊　01234
捜査官 10---
理由 捜査官 1 は捜査官 0 より先に出発できない

不可能な配置の例 (2)

盗賊　01234
捜査官 --0--
理由 捜査官 1 も配置しなければならない

1. M=2 の場合、つまり 2 人の捜査官を配置する状況を考える。回収額の最大値を計算する関数 f2 を完成させるために、以下のプログラム中の空欄 (A) ~ (D) を埋めよ。

```
def f2()
  max = 0
  for i in 0..N-2
    rcvr_i = (A)  # 初めの捜査官を i 番目の盗賊に配置して回収できる金額
    for j in (B)..N-1
      rcvr_j = (C)  # 次の捜査官を j 番目の盗賊に配置して回収できる金額
      rcvr = rcvr_i + rcvr_j
      if max < rcvr
        max = rcvr
      end
    end
  end
  (D)
end
```

2. M=3 での最大値を求める関数 f3 を前問と同様に全ての配置を調べるという方針で作った場合の計算量のオーダーを、変数 N を用いて表わせ。
3. 一般の N,M について、動的計画法を用いた効率的な解法を設計したい。元の問題の一部 (または全部) として、先頭から n 人の盗賊に先頭から m 人の捜査官を配置した時の最大回収額を rcvr[n][m] で表記するとする。

$$\texttt{rcvr[n][m]} = \begin{cases} 0 & m = 0 \text{ または } n = 0 \text{ または } m > n \\ \max(\texttt{rcvr[n-1][m]}, \boxed{\text{(E)}}) & \text{それ以外の場合} \end{cases}$$

空欄 (E) を埋めて漸化式を完成させよ。ただし空欄 (E) は n-1 番目の盗賊に m-1 番目の捜査官を配置した場合に対応する。また $0 \leq m \leq$ M, $0 \leq n \leq$ N の範囲のみ考えるとする.

4. この方針で問題を解く関数 g を、下記の空欄を埋める形で完成させよ。インデント (行頭の空白) を適切にとること。

```
def g()
  rcvr = make2d(N+1,M+1)
  ... ここを埋める (複数行可)...
  rcvr[N][M]
end
```