付 録A　演習問題のヒントと解答例（補足分）

【解答例 1.1】　何を出力するか？（１）
　以下のようになります.

```
x = 3   y = 1
x = 1   y = 3
x = 0   y = 0
x = 0   y = 0
i = 28  l = 28  f = 28  d = 28
i = 28  l = 28  f = 28  d = 28
i = 28  l = 28  f = 28.571428   d = 28.571429
i = 28  l = 28  f = 28  d = 28
i = 28571       l = 28571       f = 28571       d = 28571
i = 28571       l = 28571       f = 28571       d = 28571
```

【解答例 1.2】　何を出力するか？（２）
　以下のようになります.

```
x=5     y=5     z=5
x=1     y=5     z=5
x=1     y=5     z=5
 x < y ? x ++ : y ++  = 6
x=9     y=7     z=3
x=2     y=1     z=1
x=0     y=-1    z=-1
```

【解答例 1.6】 スミス数

以下のプログラムでは，各位の数字の和を関数 get_digit_sum で計算しています．なおこの関数は再帰（2 章参照）を用いても書くことができます．また素因数分解をして桁の和をとる必要もあります．これは

1. まず元の数が奇数になるまで 2 で割り算をする．
2. その後，3 から奇数を順番に 3,5,7,9……と使って割り切れなくなるか，もしくは 1 になるまでその数で割る．
3. それぞれの素因数の桁の和を get_digit_sum を用いて求め，sum に加える．

という操作を関数 get_factor_digit_sum によって実現しています．数 n がスミス数であるかどうかの判定は，get_digit_sum(n)==get_factor_digit_sum(n) によってできます．ただし素数を除くため，get_factor_digit_sum 関数内において素因数の数をカウントし，素数の場合には 0 を返すようにしています．これにより素数は get_digit_sum 関数の返り値と一致することはなく，スミス数の解から排除されます．

【プログラム】 A.1 スミス数を計算するプログラム

```
#include<stdio.h>
#include<stdlib.h>

/* 整数の各桁の和を求める関数 */
int get_digit_sum(long num)
{
  int sum = 0;
  while(num>0){
    sum += num % 10; /* 最下位の数字を sum に加算する */
    num /= 10;       /* num から最下位の数字を除いた数に変換 */
  }
  return sum;
}
```

```
/* 素因数分解して，各桁の和を求める関数 */
int get_factor_digit_sum(long num)
{
  int sum = 0;          /* 最終的に解を返すための変数 */
  int i;
  int num_factor = 0;  /* 素数を除外するために素因素の数を保存する */
  /* まずは偶数でなくなるまで 2 で割り続ける */
  while(num % 2==0){
    num /= 2;
    sum += 2;
    num_factor++;
  }
  /* その後，num が 1 になるまで奇数で割り続ける */
  for(i= 3; num!= 1 ;i+= 2){
    while(num % i==0){
      num /= i;
      sum += get_digit_sum(i); /* 見つけた素因数の各桁の和を求める必
要がある */
      num_factor++;
    }
  }
  /* 素数だった場合は 0 を返して後の判定で自動的に除外されるようにする */
  if(num_factor != 1) return sum;
  else return 0;
}

main(int argc,char **argv)
{
  long num,max=0;
  int digit_sum,factor_sum;
  /* 探索の最大値はプログラムの引数で指定する */
  /* 指定されていない場合は終了する */
  if(argc != 2){
    printf("usage:%s (MAX_NUMBER)\n",argv[0]);
```

```
    exit(0);
  }
  max = atoi(argv[1]);  /* 引数は文字列なので整数に変換して保存する.
atoi() は stdlib.h にある標準ライブラリ関数 */
  for (num=2; num <= max; num++){
    digit_sum = get_digit_sum(num); /* 各桁の和を求める */
    factor_sum = get_factor_digit_sum(num); /* 素因数分解して，各桁
の和を求める */
    if(digit_sum == factor_sum ) /* 素数は factor_sum==0 になるので除
外される */
      printf("%ld\n", num);
  }
  return 0;
}
```

【出力例】 A.1 スミス数の表示

```
iba@fs(~/tmp)[521]: ./a.out 200
4
22
27
58
85
94
121
166
```

このプログラムは起動に与えた引数以下の整数に対してスミス数を探索します．大きな数を入力すると実行にかなりの時間がかかります．これは素因数分解の計算量が大きいからです．より短時間で多くのスミス数を発見するには，前もって素数のリストを作っておくとよいでしょう．これにより因数分解の無駄な計算を減らすことができます．なお，素因数分解のより効率的なアルゴリズムも近年報告されています．

【解答例 1.8】 ウェアリングの問題

以下のプログラムは再帰を用いて平方数の和をすべて表示するものです. 任意の乗数の和についても，同じようにしてプログラムを作成できます.

【プログラム】 A.2 ウェアリングの問題（平方数）の解を計算するプログラム

```
#include <stdio.h>
#define MAXL 4

find2(int k, int n, int res[])
{
  int i,j;
  int res2[MAXL];
  for (i=1;i*i<=k;i++) {
    if (i*i==k) {
      for (j=0;j<MAXL-n;j++)
        printf("%d*%d+",res[j],res[j]);
      printf("%d*%d\n",i,i);
      return;
    }
    if (n>=2) {
      for (j=0;j<MAXL-n;j++) res2[j]=res[j];
      res2[MAXL-n]=i;
      find2(k-i*i,n-1,res2);
    }
  }
}

main()
{
  int k;
  int result[MAXL];
  printf("number?\n");
  scanf("%d",&k);
```

```
  find2(k,4,result);
}
```

【出力例】 A.2 和の表示

```
iba@fs(~/tmp)[719]: a.out
number?
23
1*1+2*2+3*3+3*3
1*1+3*3+2*2+3*3
1*1+3*3+3*3+2*2
2*2+1*1+3*3+3*3
2*2+3*3+1*1+3*3
2*2+3*3+3*3+1*1
3*3+1*1+2*2+3*3
3*3+1*1+3*3+2*2
3*3+2*2+1*1+3*3
3*3+2*2+3*3+1*1
3*3+3*3+1*1+2*2
3*3+3*3+2*2+1*1
```

立法数や 4 乗数も同じようにして作成できます.

【解答例 2.1】 6174 の不思議

1111,2222,...,9999 以外のどのような 4 桁の数に対しても，g の操作を続けていくと 6174 になります.

【プログラム】 A.3 $g(g\cdots g(g(x))$ を計算するプログラム

```
#include <stdio.h>

main()
{
  int x;
```

```
  scanf("%d", &x);//数字を入力
  printf(" MAX    MIN\n");
  g(x);
}

g(int x)
{
  int d[4], i, j, tmp, max, min;
  //得た数字の各位の数字を配列に格納する
  d[0] = x % 10;//１の位
  d[1] = (x/10) % 10;//１０の位
  d[2] = (x/100) % 10;//１００の位
  d[3] = (x/1000) % 10;//１０００の位
  //配列をソートし，d[0]<=d[1]<=d[2]<=d[3]  となるようにする
  for(i=0; i<=2; i++)
    for(j=i+1; j<=3; j++)
      if(d[i] > d[j]){
        tmp = d[i]; d[i] = d[j]; d[j] = tmp;
      }
  //最大，最小の数字をもとめ，差をだす
  max = 1000*d[3] + 100*d[2] + 10*d[1] + d[0];
  min = 1000*d[0] + 100*d[1] + 10*d[2] + d[3];
  printf("%4d - %4d = %4d\n", max, min, max-min);
  if(x == max - min) return; // MIX==MAX の数となったので終了
  g(max - min); // g(x) の再帰的呼び出し
}
```

このプログラムでは関数 g(x) の中で最後に自分自身をに再帰的に呼びだしています．後述するようにどのような４桁の数に対しても max==min となる数に至ることが分かっているので，無限ループに陥ることはなくプログラムは終了します．

【出力例】 A.3 ６１７４に至る変化の様子

```
iba@fs(~/tmp)[622]: ./a.out
```

```
124
 MAX    MIN
4210 -  124 = 4086
8640 -  468 = 8172
8721 - 1278 = 7443
7443 - 3447 = 3996
9963 - 3699 = 6264
6642 - 2466 = 4176
7641 - 1467 = 6174
7641 - 1467 = 6174
iba@fs(~/tmp)[623]: ./a.out
3472
 MAX    MIN
7432 - 2347 = 5085
8550 -  558 = 7992
9972 - 2799 = 7173
7731 - 1377 = 6354
6543 - 3456 = 3087
8730 -  378 = 8352
8532 - 2358 = 6174
7641 - 1467 = 6174
```

なお，g の操作を続けても変わらない数が 6174 のみであることは，以下のようにして示せます．４桁の数 $abcd$ を

$$9 \geq a \geq b \geq c \geq d \geq 0 \tag{A.1}$$

とします．ここで a, b, c, d についてはすべてが同じということはないとします．４桁の数は $1111, 2222, \cdots, 9999$ ではないので $a < d$ です．このとき $abcd$ に g の操作を行って，

$$\begin{array}{cccccc} & a & b & c & d \\ - & d & c & b & a \\ \hline & A & B & C & D \end{array}$$

となったとしましょう．ここで，

$$
\begin{aligned}
D &= 10+d-a \ \ (a>d \text{ より}) && \text{(A.2)}\\
C &= 10+(c-1)-b=9+c-b \ \ (b>c-1 \text{ より}) && \text{(A.3)}\\
B &= (b-1)-c \ \ (b>c \text{ より}) && \text{(A.4)}\\
A &= a-d && \text{(A.5)}
\end{aligned}
$$

であることが分かります．C, B の計算の際には，下の桁からの繰り下がりがあるので $(c-1)$ や $(b-1)$ となっています．なお B, A については $b=c$ のときには

$$
\begin{aligned}
B &= 10+(b-1)-c\\
A &= (a-1)-d
\end{aligned}
$$

となりますが，このときは題意を満たすものが存在しません．

以上から，式(A.2)〜(A.5)を場合分けによって解くと，集合として $\{A,B,C,D\}=\{a,b,c,d\}$ となるのは $a=7, b=6, c=4, d=1$ のときのみです．したがって，$ABCD=6174$ となると，あとの g の操作で同じ数字が繰り返されることが分かります．

またどんな4桁の数（$1111,\cdots,9999$ を除く）も g の操作で必ず 6174 に至ることは，g の操作で現れる数（高々90通り，重複をうまく減らすと約3分の1になる）についてしらみつぶしに調べることで証明できます．

【解答例 2.2】　最大公約多項式

最大公約数を求めるユークリッドの互除法をもとにしたアルゴリズムによって，最大公約多項式を求めることができます．なお，計算の途中で負の数を含む最大公約数や最小公倍数を求めることもあります．その場合には正の値に変換して求めます．たとえば–12と–18の最大公約数は6，最小公倍数は36です．

以上からアルゴリズムは次のようになります．

Step1 入力を $p(x)$ および $q(x)$ とする．

Step2 $q(x) = 0$ なら $p(x)$ を出力して終了．

Step3 $p(x) := q(x)$ および $q(x) := q(x)$ を $p(x)$ で割った余り，として **Step2** へ．

ただし，最大公約多項式としては整数係数の多項式を求めるので，分数係数を避けて割り算をします．そのため，**Step3** の割り算の余りは次のような手順で求めます．

Step3 $p(x), q(x)$ のそれぞれの次数を m, n （$m \geq n$）とし，最高次の係数を a, b とする．また a と b の最小公倍数を L とする．このとき，

$$r(x) = \frac{L}{a} \times p(x) - \frac{L}{b} \times q(x) \times x^{m-n} \tag{A.6}$$

を計算し，$p(x) := q(x)$ および $q(x) := r(x)$ として **Step2** へ．

この方法で，最大公約多項式を求めると定数倍係数のものが出力されることがあります．たとえば，

$$p_2(x) = 2x^2 + 3x + 1 = (2x+1)(x+1) \tag{A.7}$$

$$q_2(x) = x^2 + 3x + 2 = (x+2)(x+1) \tag{A.8}$$

に対して，上のアルゴリズムでは $-3x - 3$ が最大公約多項式として求められます．そこで **Step2** を次のように修正します．

Step2 $q(x) = 0$ なら $p(x)$ を出力して終了．ただし，$p(x)$ の全係数の最大公約数を求め，各係数を最大公約数で割り算し，最大次の係数が負のときは -1 をかけた多項式を出力する．

【プログラム】 A.4 最大公約多項式を表示するプログラム

```
#include <stdio.h>
```

```
#include <math.h>
#define SIZE 100

/*
多項式を扱うとき，p[0] に次数を格納し，p[1] に定数項を
p[2] に x の係数を，
 ........
p[k] に x^k-1 の係数を格納する．*/

void cp_poly(int p[],int q[]); /*p の配列を q にコピーする*/
void normalize_poly(int p[]);  /*多項式が 2x + 4 のような場合，2 で割り，多項式 x+2 を p に代入する関数*/
void mul_poly(int p[],int a);  /*多項式に整数をかける関数．結果は p に格納．*/
void shift_poly(int p[],int n); /*多項式に x^n をかける関数．掛け算の結果を p に格納．*/
void minus_poly(int p[],int q[]); /*多項式の引き算をする関数．p-q をする．結果は p に格納する．ただし，p の次数 >= q の次数とする．*/
void div_poly(int p[],int q[],int rem[]); /*多項式の割り算 p/q をする関数．再帰を行う．余りは rem に格納．*/
void gcd_poly(int p[],int q[],int gcd[]); /*最大公約多項式を求める関数．最大公約多項式を gcd に格納する．*/
void print_poly(int p[]); /*多項式を表示する関数*/
int gcd(int m,int n);     /*最大公約数を求める関数*/
int lcm(int m,int n);     /*最小公倍数を求める関数*/

void cp_poly(int p[],int q[])
{
  int i;
  for(i = 0; i < SIZE;i++) q[i] = p[i];
}

void normalize_poly(int p[])
{
  int i;
```

```
  int G ;
  /*pが 0 次式なら多項式 1 とする*/
  if(p[0] == 0) p[1] = 1;
  else{
    /*p の係数の最大公約数を求める*/
    G = gcd(p[1],p[2]);
    for(i = 3;i <= p[0]+1;i++){
      G = gcd(p[i],G);
    }
    /*最大次の係数が正の時は各係数を G で割る*/
    if(p[p[0]+1] > 0)
      for(i = 1; i <= p[0]+1;i++) p[i] = p[i]/G;
    /*最大次の係数が負の時は各係数を G で割った結果に-1 をかける*/
    else if(p[p[0]+1] < 0)
      for(i = 1; i <= p[0]+1;i++) p[i] = -1 * p[i]/G;
  }
}

void mul_poly(int p[],int a)
{
  int i;
  /*a = 0 の時は 0 を返す*/
  if(a == 0){
    p[0] = 0;       /*次数を 0 とする*/
    p[1] = 0;
  }
  else for(i = 1;i <= p[0]+1;i++) p[i] = a*p[i];
}

void shift_poly(int p[],int n)
{
  int i;
  int tmp[SIZE];
  tmp[0] = p[0]+n;
  for(i = 1;i <= n;i++) tmp[i] = 0;
```

```
  for(i = n+1;i <= p[0]+1+n;i++) tmp[i] = p[i-n];
  cp_poly(tmp,p);
}

void minus_poly(int p[],int q[])
{
  int i;
  int degree = 0;
  for(i = 1;i <= p[0]+1 && i <= q[0]+1;i++){
    p[i] = p[i] - q[i];
  }
  /*引き算を行うと次数が変わる可能性があるので次数を調べる*/
  for(i = 1;i <= p[0]+1 || i <= q[0]+1;i++){
    if(p[i] != 0) degree = i-1;
  }
  p[0] = degree;
}

void div_poly(int p[],int q[],int rem[])
{
  int L;
  int tmp_q[SIZE]; /*qをコピーを保存する配列*/
  if(p[0] < q[0]) cp_poly(p,rem);
  else if(p[0] == 0 && p[1] == 0){
    rem[0] = 0;
    rem[1] = 0;
  }
  else {
    L = lcm(p[p[0]+1],q[q[0]+1]);
    cp_poly(q,tmp_q);
    mul_poly(p,L/p[p[0]+1]);
    mul_poly(tmp_q,L/tmp_q[tmp_q[0]+1]);
    shift_poly(tmp_q,p[0] - tmp_q[0]);
    minus_poly(p,tmp_q);
    div_poly(p,q,rem);
```

```
  }
}

void gcd_poly(int p[],int q[],int gcd[])
{
  int rem[SIZE];
  if(q[0] == 0 && q[1] == 0){
    cp_poly(p,gcd);
    normalize_poly(gcd);
  }
  else {
    div_poly(p,q,rem);
    gcd_poly(q,rem,gcd);
  }
}

void print_poly(int p[])
{
  int i;
  /*次数の大きい方から 3x^n + 4x^n-1 + ... のように表示*/
  for(i = p[0];i > 1;i--){
    if (p[i+1]==0) continue;
    if (p[i+1]==1) printf("x^%d + ",i);
    else printf("%dx^%d + ",p[i+1],i);
  }
  if (p[2]!=0)
     if (p[2]==1) printf("x + ",p[2]);
     else printf("%dx + ",p[2]);
  if (p[1]!=0) printf("%d\n",p[1]);
  return;
}

int gcd(int m,int n)
{
  int ret;
```

```
  m = abs(m);
  n = abs(n);
  if(n == 0) ret = m;
  else ret = gcd(n,m%n);
  return ret;
}

int lcm(int m,int n)
{
  m = abs(m);
  n = abs(n);
  int G = gcd(m,n);
  return n/G*m;
}

main()
{
  int p[SIZE],q[SIZE],gcd[SIZE];
  int i;

  /*多項式 p と q の入力*/
  /*p の引数の入力*/
  printf("p degree = ");
  scanf("%d",&p[0]);
  /*p の係数の入力*/
  puts("input p factor");
  for(i = 0;i <= p[0];i++){
    printf("a_%d = ",i);
    scanf("%d",&p[p[0]+1-i]);
  }
  if(p[p[0]+1] == 0){            /*最高次の係数が 0 ならエラー*/
    printf("invalid input.\n");
    return 0;
  }
  /*q の引数の入力*/
```

```
  printf("q degree = ");
  scanf("%d",&q[0]);
  /*q の係数の入力*/
  puts("input q factor");
  for(i = 0;i <= q[0];i++){
    printf("b_%d = ",i);
    scanf("%d",&q[q[0]+1-i]);
  }
  if(q[q[0]+1] == 0){              /*最高次の係数が 0 ならエラー*/
    printf("invalid inpu.t\n");
    return 0;
  }
  /*入力した多項式の表示*/
  printf("p(x) = ");
  print_poly(p);
  printf("q(x) = ");
  print_poly(q);

  gcd_poly(p,q,gcd);
  /*最大公約多項式の表示*/
  printf("gcd(x) = ");
  print_poly(gcd);
  return 0;
}
```

【出力例】 A.4 最大公約多項式の計算例

```
iba@fs(~/tmp/1998_3_3)[540]: ./a.out
p degree = 3
input p factor
a_0 = 1
a_1 = 5
a_2 = 7
a_3 = 3
```

```
q degree = 2
input q factor
b_0 = 1
b_1 = 4
b_2 = 3
p(x) = x^3 + 5x^2 + 7x + 3
q(x) = x^2 + 4x + 3
gcd(x) = x^2 + 4x + 3
```

なおこのプログラムではもとの $p(x)$ や $q(x)$ の係数が互いに素でない場合には簡約した式を出力します．たとえば，

$$p(x) = (2x+2)(x+3) \tag{A.9}$$
$$q(x) = (2x+2)(x+4) \tag{A.10}$$

に対して，$x+1$ を出力します．これについては容易に修正ができるでしょう．

【解答例 2.4】 Newton-Raphson 法の 2 次の収束性

真の解 x^* で関数 $f(x)$ のテイラー展開を第 3 項まで計算すると，

$$f(x^*) = f(x_k+\epsilon_k) = f(x_k) + f'(x_k)\epsilon_k + f''(x_k)\frac{\epsilon_k^2}{2} + O(\epsilon_k^3) = 0 \tag{A.11}$$

となります．ただし $\epsilon_k = x^* - x_k$ です．この式を $f'(x_k) \neq 0$ で割ると

$$\frac{f(x_k+\epsilon_k)}{f'(x_k)} \approx \frac{f(x_k)}{f'(x_k)} + \epsilon_k + \frac{f''(x_k)\epsilon_k^2}{2f'(x_k)} = 0 \tag{A.12}$$

となり，したがって

$$\frac{f(x_k)}{f'(x_k)} + \epsilon_k = -\frac{f''(x_k)\epsilon_k^2}{2f'(x_k)} \tag{A.13}$$

を得ます．これと式 (2.8) を用いると，

$$x^* - x_{k+1} = -(x^* - x_k)^2 \times \frac{f''(x_k)}{2f'(x_k)} \tag{A.14}$$

となります．ここで $|\frac{f''(x_k)}{f'(x_k)}|$ の最大値を $2C$ とおくと

$$|x^* - x_{k+1}| \leq \frac{(x^* - x_k)^2 \times \quad 2C}{2} = C|x_k - x^*|^2 \qquad \text{(A.15)}$$

です．

【解答例 2.7】 カエルパズル (2)

関数 S(n,s) を以下のように書き換えればよいことが分かります．

【プログラム】 A.5 カエルパズルのプログラム（２）

```
void S(int n, char *s)
{
  char news1[MAX_N],news2[MAX_N];

  if (n==1)
    {
      printf("%s%s\n", s,"* B");
      return;
    }
  if (n==2)
    {
      printf("%s%s\n", s,"* * B");
      printf("%s%s\n", s,"_ _ B");
      return;
    }
  S(n-2, strcat(strcpy(news1,s),"_ _ "));
  S(n-1, strcat(strcpy(news2,s),"* "));
}
```

出力例は次のようになります．

【出力例】 A.5 カエルパズルの出力例（２）

```
n=4
```

```
A _ _ * * B
A _ _ _ _ B
A * _ _ * B
A * * * * B
A * * _ _ B
```

【解答例 3.1】 乱数で自然対数の底をつくる (1)

以下のようになります.

【プログラム】 A.6 *M/Z*を求めるプログラム

```
#include <stdio.h>
#include<stdlib.h>
#define MMAX 10000
main()
{
    int seed;
    int j,k,n;    //ループ用カウンタ
    int m;      //M
    int box[MMAX];  //箱
    float x;    //乱数
    float min,max;  //箱の境界
    int z;      //Z
    printf("input seed = ");
    scanf("%d",&seed);
    srand(seed);
    printf("input M = ");
    scanf("%d",&m);
    z=m;
    //箱の初期化
    for (j=0; j<m; j++) box[j]=0;
    for (k=0; k<m; k++) {
        x = rand();
        x = m*(x/RAND_MAX); //x を０〜M の範囲の乱数にする
```

```
        for (n=0; n<m; n++) {
        //x を長さ 1 の箱に入れ，何も入っていない箱だったら Z から 1 を引く
            min=n;
            max=n+1;
            if( (x>=min) && (x<=max) ) {
                if(box[n]==0) {
                    box[n]=1;
                    z=z-1;
                }
            }
        }
    }
    printf("M/Z = %f\n",(float)((float)m/(float)z));
}
```

このプログラムでは次のことを実行します.

Step1 ランダムシードと試行回数 M を入力させる.

Step2 Z=M とする.

Step3 最初は何も入っていないので，この状態を box[j]=0（j=1,2,,M）とする.

Step4 0〜M の範囲の乱数 x を作る..

Step5 これを [0,1][1,2][2,3][3,4][4,5]...[M-1,M] という M 個の箱に入れると想定し，入れた箱の box を 1 とする.

Step6 あらたに値が 1 となる box があれば，Z から 1 を引く.

Step7 **Step3**〜**Step5** を M 回くりかえす.

Step8 M/Z を求める.

【出力例】 A.6 *M/Z*の出力例

```
iba@fs(~/tmp)[521]: ./a.out
input seed = 123
input M = 1000
M/Z = 2.659574
iba@fs(~/tmp)[522]: ./a.out
input seed = 123
input M = 10000
M/Z = 2.648305
```

【解答例 3.2】 乱数で自然対数の底をつくる (2)

（１）$S_n = X_1 + X_2 + \cdots + X_n$ と書くと，任意の n について，

$$\begin{aligned} P(N > n) &= P(S_1 < 1, S_2 < 1, \cdots, S_n < 1) \\ &= P(X_1 < 1, X_1 + X_2 < 1, \cdots, X_1 + X_2 + \cdots + X_n < 1) \\ &= P(X_1 + X_2 + \cdots + X_n < 1) \end{aligned}$$

となります．最後の等式は，nが大きくなるとともに和が単調増加することによります．次に

$$\begin{aligned} P(N = n) &= P(N > n-1) - P(N > n) \\ &= P(S_{n-1} < 1) - P(S_n < 1) \end{aligned}$$

となることから，

$$\begin{aligned} E(N) &= \sum_{n=2}^{\infty} nP(N = n) \\ &= \sum_{n=2}^{\infty} n[P(S_{n-1} < 1) - P(S_n < 1)] \\ &= 2[P(S_1 < 1) - P(S_2 < 1)] + 3[P(S_2 < 1) - P(S_3 < 1)] \end{aligned}$$

$$
\begin{aligned}
& +4[P(S_3<1)-P(S_4<1)]+\cdots \\
= \;& 2P(S_1<1)+P(S_2<1)+P(S_3<1)+\cdots \\
= \;& 2+P(S_2<1)+P(S_3<1)+\cdots
\end{aligned}
$$

です（ただし $P(S_1<1)=1$ に注意）.

$$
P(S_n<1)=\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\cdots\int_0^{1-x_1-x_2-\cdots-x_{n-1}} dx_n dx_{n-1}\cdots dx_2 dx_1 \tag{A.16}
$$

から数学的帰納法により $P(S_n<1)=\frac{1}{n!}$ を導くことができます（これは n 次元のハイパー体積と呼ばれている）. したがって

$$
E(N)=2+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots\frac{1}{n!}+\cdots
$$

となり，$E(N)=e$ が分かります.

（２）１つの乱数が特定の区間（M 個の箱のうちの１つ）に入る確率は $1/M$ です. したがって，M 個の乱数を発生させた場合にどれもがこの区間に入らない確率は $P(0)=(1-\frac{1}{M})^M$ となります. この値は M が大きくなると $1/e$ に近づきます. Z/M は $P(0)$ の概算値になることから，$e\approx\frac{M}{Z}$ となります.

【解答例 3.4】　ビュフォンの針 (2)

針の長さを K，線の幅を L とします. 針と平行線の対称性から，針は x 軸からの傾き $\theta(0\le\theta\le\pi/2)$ で $0\le y\le L$ に落ちるとしていいでしょう. 傾き θ で落ちた針は $y\le K\sin\theta$ のときに平行線と交わります. このとき K と L の大小で場合わけします.

- $K\le L$ のとき
  傾き θ で落ちた針は $P_\theta=K\sin\theta/L$ の確率で交わります. よって，針が平行線と交わる確率は，$P=\frac{\int_0^{\pi/2}K\sin\theta/Ld\theta}{\frac{\pi}{2}}=\frac{2K}{\pi L}$ となります.

- $K > L$ のとき
  $\sin\theta \geq K/L$ のときは，傾き θ で落ちた針の交わる確率は $P_\theta = 1$ となります．それ以外の場合は，$P_\theta = K\sin\theta/L$ の確率で交わります．したがって，針が平行線と交わる確率は，

$$P = \{(\int_0^{\varphi} K\sin\theta/L d\theta) + (\int_{\varphi}^{\pi/2} d\theta)\}/(\frac{\pi}{2}) \quad \text{(A.17)}$$

$$= \frac{2}{L\pi}\left(K - \sqrt{K^2 - L^2}\right) + \left(1 - \psi\cdot\frac{2}{\pi}\right) \quad \text{(A.18)}$$

  となります．ただし $\sin\psi = K/L$ $(0 < \psi < \frac{\pi}{2})$ です．

方眼紙のように線が縦横に（等間隔 L で）引かれている場合を考えましょう．とくに $K \leq L$ のときには，傾き θ で落ちた針は $x \leq K\cos\theta$ かつ $y \leq K\sin\theta$ のときに平行線と交わりません．つまり，傾き θ で落ちた針は $P_\theta = 1 - \left(1 - \frac{K}{L}\cos\theta\right)\cdot\left(1 - \frac{K}{L}\sin\theta\right)$ の確率で平行線と交わります．よって，針が平行線と交わる確率は $P = \frac{\int_0^{\pi/2} P_\theta \quad d\theta}{\frac{\pi}{2}} = \frac{K}{\pi L}\cdot\left(4 - \frac{K}{L}\right)$ です．とくに $K = L$ のとき $P = \frac{\pi}{3}$ となります．

【解答例 4.1】　何を出力するか？(3)
次のようになります．その理由を考えて見ましょう．

```
IBA
IBA
BA
IA
ABI
ABI
ABI
```

【解答例 5.1】　何を出力するか？(4)
以下のようになります．その理由を考えてみましょう．

```
a=134518604 *a=0
p=134518620 *p=134518604 **p=0
pp=134518620 *pp=134518604 **pp=0
pp-p=1 *pp-a=1 **pp=1
pp-p=2 *pp-a=2 **pp=2
pp-p=2 *pp-a=3 **pp=3
pp-p=1 *pp-a=1 **pp=1
pp-p=1 *pp-a=2 **pp=2
```

【解答例 5.2】　何を出力するか？(5)
　以下のようになります．その理由を考えてみましょう．

```
a[i][1-i]=2 *a[i]=1 *(*(a+i)+i)=1
a[i][1-i]=3 *a[i]=3 *(*(a+i)+i)=4
*pa[i]=1 p[i]=1
*pa[i]=3 p[i]=2
```

【解答例 6.1】　何を出力するか？(6)
　以下のようになります．その理由を考えてみましょう．

```
Kohki Kohki Hiroto
1999
2002
2004
irono Kohki Kohki
```

【解答例 7.6】　Lisp のインタプリタ (1)

【プログラム】 A.7 *LISP* のインタプリタ

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
```

```
#include <math.h>
#define MAXNAME 128
#define MAXLENGTH 128
#define MAX_ARGS 10
#define INTEGER 1
#define DOUBLE 2
#define TRUE 3
#define FALSE 4

/* 木構造を作るノードの構造体 */
struct node
{
  int no_of_children;
  struct node *children[MAX_ARGS];
  char name[MAXNAME];
};

/* 値を格納する構造体 */
struct value
{
  int type;
  int ikata;
  double dkata;
  char ckata[MAXNAME];
};

/* 文字を取り込む関数 */
void get_next_token(char *buf)
{
  int   ibuf;
  int   i = 0;
  while ((ibuf = getchar()) != EOF) {
    if (isspace((char) ibuf)) {
      buf[i] = '\0';
      break;
```

```
    } else if (((char) ibuf) == ')') {
      ungetc((char) ibuf,stdin);
      buf[i] = '\0';
      break;
    } else {
      buf[i++] = (char) ibuf;
    }
  }
}


/* 木構造を組み立てる関数 */
struct node *read_tree()
{
  int   i,j;
  struct node *t;
  char  buf[MAXLENGTH];
  if (scanf("%1s",buf) == EOF) return (struct node *) EOF;
  if ((t = (struct node *) malloc(sizeof(struct node))) == NULL) {
    printf("Error: memory overflow\n");
    exit(1);
  }
  t -> no_of_children=0;
  if (buf[0] == '(') {
    scanf("%1s", buf);
    ungetc(buf[0],stdin);
    get_next_token(buf);
    strcpy(t->name,buf);
    scanf("%1s",buf);
    for(i = 0; buf[0] != ')'; scanf("%1s",buf)) {
      ungetc(buf[0],stdin);
      t->children[i++] = read_tree();
    }
    t->no_of_children=i;
    ungetc(buf[0], stdin);
```

```
    scanf(")");
  } else if (buf[0] == ')') {     /* ')'ならポインタをNULLにする */
    scanf(")");
    return NULL;
  } else {
    ungetc(buf[0],stdin);
    get_next_token(buf);
    strcpy(t->name,buf);
    t->children[0] = NULL;
  }
  return(t);
}

/* 木構造をプリントする関数. デバッグ用 */
void print_tree(struct node *nd)
{
  int j;
  if (nd->no_of_children == 0){
    printf(" %s",nd->name);
    return;
  } else {
    printf(" (%s",nd->name);
    for(j=0;j<nd->no_of_children; j++)
      print_tree(nd->children[j]);
    printf(")");
  }
}

/* スペースを解放する関数 */
void free_tree(struct node *nd)
{
  int j;
  for (j=0;j < nd->no_of_children;j++)
    free_tree(nd->children[j]);
  free(nd);
```

```
}

/* 値を評価する関数 */
struct value *eval_tree(struct node *nd, struct value *val)
{
  int j,k,i;
  double l,r,num,num2;
  struct value child;
  char *p;
  // 以下はデバッグ用のプリント
  //  printf("%s %d\n", nd->name, nd->no_of_children);
  // 終端ノードの場合
  if (nd->no_of_children == 0){
    if (*nd->name == '-' || isdigit(*nd->name)){  /* 数値の場合 */
      val->ikata = atoi(nd->name);     /* 数値を格納する構造体に型別
に格納 */
      val->dkata = atof(nd->name);
      val->type = INTEGER;
      p = nd->name;
      while(*p != '\0')
          if (*p++ == '.') val->type = DOUBLE;
    }
    else printf("Undefined symbol : %s\n",nd->name);
  }
  // 関数ノードの場合
  else {
    // + 演算子
    if (strcmp(nd->name,"+") == 0){
      val->ikata = 0;
      val->dkata = 0;
      val->type = INTEGER;
      for (j=0;j<nd->no_of_children;j++){
           eval_tree(nd->children[j],&child);
           // 以下はデバッグ用のプリント
           // printf("%f\n", child.dkata);
```

```
        val->ikata += child.ikata;
        val->dkata += child.dkata;
        if (child.type == DOUBLE) val->type = DOUBLE;
    }
    return;
  }
  // * 演算子
  if (strcmp(nd->name,"*") == 0){
    val->ikata = 1;
    val->dkata = 1;
    val->type = INTEGER;
    for (j=0;j<nd->no_of_children;j++){
        eval_tree(nd->children[j],&child);
        val->ikata *= child.ikata;
        val->dkata *= child.dkata;
        if (child.type == DOUBLE) val->type = DOUBLE;
    }
    return;
  }
  // - 演算子
  if (strcmp(nd->name,"-") == 0){
    eval_tree(nd->children[0],&child);
    val->ikata = child.ikata;
    val->dkata = child.dkata;
    val->type = child.type;
    for (j=1;j<nd->no_of_children;j++){
        eval_tree(nd->children[j],&child);
        val->ikata -= child.ikata;
        val->dkata -= child.dkata;
        if (child.type == DOUBLE) val->type = DOUBLE;
    }
    return;
  }
  // / 演算子
  if (strcmp(nd->name,"/") == 0){
```

```
    eval_tree(nd->children[0],&child);
    val->ikata = child.ikata;
    val->dkata = child.dkata;
    val->type = child.type;
    for (j=1;j<nd->no_of_children;j++){
         eval_tree(nd->children[j],&child);
         val->ikata /= child.ikata;
         val->dkata /= child.dkata;
         if (child.type == DOUBLE) val->type = DOUBLE;
    }
    return;
  }
  // cos 関数
  if (strcmp(nd->name,"COS") == 0){
    val->dkata = 0;
    val->type = DOUBLE;
    eval_tree(nd->children[0],&child);
    val->dkata = cos(child.dkata);
    return;
  }
  // sin 関数
  if (strcmp(nd->name,"SIN") == 0){
    val->dkata = 0;
    val->type = DOUBLE;
    eval_tree(nd->children[0],&child);
    val->dkata = sin(child.dkata);
    return;
  }
  // < 比較演算子
  if (strcmp(nd->name,"<") == 0){
    eval_tree(nd->children[0],&child);
    l = child.dkata;
    eval_tree(nd->children[1],&child);
    r = child.dkata;
    if (l < r) val->type = TRUE;
```

```
  else val->type = FALSE;
  return;
}
// > 比較演算子
if (strcmp(nd->name,">") == 0){
  eval_tree(nd->children[0],&child);
  l = child.dkata;
  eval_tree(nd->children[1],&child);
  r = child.dkata;
  if (l > r) val->type = TRUE;
  else val->type = FALSE;
  return;
}
// eq 演算子
if (strcmp(nd->name,"EQ") == 0){
  k = 0;
  eval_tree(nd->children[0],&child);
  num = child.dkata;
  for (j=0;j<nd->no_of_children;j++){
       eval_tree(nd->children[j],&child);
       if (num == child.dkata) k++;
  }
  if (k == nd->no_of_children) val->type = TRUE;
  else val->type = FALSE;
  return;
}
// AND 演算子
if (strcmp(nd->name,"AND") == 0){
  val->type = TRUE;
  for (j=0;j<nd->no_of_children;j++){
       eval_tree(nd->children[j],&child);
       if (child.type != TRUE) {
            val->type = FALSE;
            break;
       }
```

```
      }
      return;
    }
    // OR 演算子
    if (strcmp(nd->name,"OR") == 0){
      val->type = FALSE;
      for (j=0;j<nd->no_of_children;j++){
           eval_tree(nd->children[j],&child);
        if (child.type == TRUE) {
          val->type = TRUE;
          break;
        }
      }
      return;
    }
    // undefined symbol
    printf("Undefined function : %s\n",nd->name);
    return;
  }
}

/* main 関数 */
main()
{
  struct node *root;
  int ct,i;
  struct value val;
  while(1){
    // プロンプトを表示し，S 式を読み込む
    printf(">");
    root = read_tree();
    // 入力された S 式を表示: デバッグ用
    // printf("--->\n");
    // print_tree(root);
    // printf("\n");
```

```
    // S 式の評価
    eval_tree(root, &val);
    switch(val.type){
    case INTEGER: printf("%d",val.ikata);break;
    case DOUBLE: printf("%f",val.dkata);break;
    case TRUE: printf("T");break;
    case FALSE: printf("NIL");break;
    }
    free_tree(root);
    printf("\n");
  }
}
```

【出力例】 A.7 数式評価の実行例

```
iba@fs(~/TEX/Kaisetsu08/ohm/shiryo/lisp)[587]: a.out
>(EQ (COS 0) 1)
T
>(+ (* 2 3) 1.9 2)
9.900000
>(SIN (+ 2 A))
Undefined symbol : A
-0.756802
>
```