Ruby Exercises

1 Calculation

1. Calculate the following.
   (a) Convert 22 degrees Celsius to Fahrenheit, where C degrees Celsius is equal to $F = \frac{9}{5}C + 32$ degrees Fahrenheit.
   (b) Convert 50 degrees Fahrenheit to Celsius.
   (c) Translate 535,800 yen (UTokyo tuition fees) to US dollars, where we suppose that 1 US dollar is 93.51 yen.
   (d) Translate 12900 US dollar (Stanford Univ. tuition fees) to Japanese yen.
2. Describe a Ruby expression to compute the following. Note that we have to use multiple functions in some cases.
   (a) $\sqrt{10}$, $(\sqrt{2}\sqrt{5})$, $\sqrt{\sqrt{5}}$, $12^{\sqrt{2}}$
   (b) $\sin 30°$, $\cos 30°$, $\tan 30°$
   (c) $\log 1000$, $\log_{100} 100$, $\log_2 1000$
   (d) 2.7^{10} (with/without using **)
3. Recalculate Problem 1 by using a variable.
4. Do the following computation using variables and assignment.
   (a) Suppose that $x = 10$, $y = x(x-3)$, $z = y(y-3)$. Compute $z(x-3)$. Moreover, describe this expression only using x by your hand.
   (b) Below is a description to compute a quadratic equation.
      i. Suppose that $a = 3$, $b = 5$, $c = -7$.
      ii. We define d as the discriminant of the quadratic equation $ax^2+bx+c = 0$.
      iii. Compute the two solutions of $ax^2+bx+c = 0$, and assign them to variables p, q.
      iv. Compute $ap^2 + bp + c$ and $aq^2 + bq + c$.

2 Functions

1. (a) Define a function f(x) that computes $2x^2 + 3x + 4$.
   (b) Define a function g(x) that computes the remainder of a given number x when divided by 5.
   (c) Define a function h() that returns "hello!" in the terminal.
2. (a) Define an original function log_3(n) that computes $\log_3(n)$, using implemented functions log(x) and/or log10(x).
   (b) Define an original function log_b(n, b) that computes $\log_b(n)$, using log(x) and/or log10(x).
3. (a) Define a function area(r) that computes the area of a circle with radius r.
   (b) Using the function area, compute the length of a regular square whose area is equal to that of a circle with radius 10cm.
   (c) Using the function area, compute the length of a regular square whose area is equal to that of a semicircle with radius 20cm.
   (d) Using the function area, compute the length of a regular square whose area is equal to that of a quarter round with radius 30cm.
4. Define the following functions.
   (a) a function triangle(x) that returns the area of a regular(equilateral) triangle xcm on side.
   (b) a function tetrahedron(x) that computes the volume of a regular tetrahedron xcm on side. Note that the height of the tetrahedron is sqrt(2/3.0)*x.
5. Define a function time_to_seconds(h,m,s) that transform "h hours m minutes s seconds" to seconds.
6. Define the following functions. In addition, give an example of using these functions. It is better to put the functions in a file whose name is given between ( ).
   (a) a function celsius_to_fahrenheit(c) that converts Celsius temperature c to Fahrenheit. (yardpound.rb)
   (b) a function fahrenheit_to_celsius(f) that converts Fahrenheit temperature f to Celsius temperatures. (yardpound.rb)
   (c) a function ms_to_mph(v) that converts a velocity v[m/s] to mile per hour(mph). (yardpound.rb)
   (d) a function mph_to_ms(v) that converts a velocity v[mph] to meter per seconds[m/s]. (yardpound.rb)
   (e) In U.S., the Wind Chill Index[$\circ^F$] is defined as

   $$35.74 + 0.6215t - 35.75(v^{0.16}) + 0.4275t(v^{0.16}),$$

   where t is Fahrenheit temperature and v is wind speed[mph]. Describe a function wind_chill_index(t, v) that computes the Wind Chill Index for a given t and v. (wci.rb)

(f) A function wind_chill_index_celsius(t, v) that computes the Wind Chill Index when t and v are given by Celsius temperature and meter per second, respectively, and the output should be given as Celsius temperature[$\circ^C$].(wci.rb)

7. Define the following functions on computing a quadratic equation $ax^2 + bx + c = 0$.(quadratic.rb)

   (a) a function det(a,b,c) that computes the discriminant.

   (b) a function solution1(a,b,c) that returns a solution $\frac{-b+\sqrt{b^2-4ac}}{2a}$. (use the function det(a,b,c))

   (c) a function solution2(a,b,c) that returns a solution $\frac{-b-\sqrt{b^2-4ac}}{2a}$. (use an auxiliary variable to represent the common part of the two solutions)

   (d) a function quadratic(a,b,c,x) that computes the function value of the quadratic function $f(x) = ax^2 + bx + c$.

8. Do the following to know behavior of local variables.

   (a) Error happens because the variable s is determined outside the function.

```
irb
def heron(a,b,c)
   sqrt(s*(s-a)*(s-b)*(s-c))
end
a=1
b=1
c=1
s=0.5*(a+b+c)
heron(a,b,c)
quit
```

   (b) Error happens because the variable s does not exist outside the function heron.

```
irb
def heron(a,b,c)
  s=0.5*(a+b+c)
  sqrt(s*(s-a)*(s-b)*(s-c))
end
heron(1,1,1)
s
quit
```

   (c) We can use variable names different from those in the function definition.

```
irb
def heron(a,b,c)
 s=0.5*(a+b+c)
 sqrt(s*(s-a)*(s-b)*(s-c))
end
t=3
```

```
u=4
v=5
heron(t,u,v)
quit
```

9. Consider the output without making a program.

```
def f(x)
  x=1
  a=2
end
a=0
f(a)
a
```

3 Conditions

1. **(absolute value)** Define a function abs(x) that computes the absolute value of a value x. (abs.rb)
2. Define a function max3(x,y,z) that returns the maximum of x,y,z.
3. Define the following functions. Note that it is better to put the functions in a file whose name is given between ( ).
   (a) a function divisible(x,y) that decides if x is divisible by y. (divisible.rb)
   (b) a function ascending(x,y,z) that decides if x < y < z. (median.rb)
   (c) a function leap_year(y) that returns true if year y is a leap year. Note that year is a leap year if it satisfies the following. (calendar.rb)
      (i) year is a leap year if it is divisible by 4, but there are exceptions as follows.
      (ii) Even when year is divisible by 4, it is not a leap year if it is divisible by 100.
      (iii) Even when year satisfies the condition of (ii), it is a leap year if it is divisible by 400.
   (d) Define a function between(a,b,x) that returns TRUE if number x is between number a and number b (including the end points), and returns FALSE otherwise.
   (e) Define a function inRectangle(a,b,c,d,x,y) that decides whether the point (x,y) is contained in the rectangle whose top left coordinate is (a,b) and bottom right is (c,d). Use the function between to define this function. It is better that the function can decide even if (a,b) is bottom left and (c,d) is top right.
4. **(variants of problem 3(d)(e))** See the next section for arrays.
   (a) a function within_range(a,i) that decides if the index i is contained in the range of the array a. Note that the index of a is starting from 0, and ends at a.length()-1. (within.rb)
   (b) a function within_image(img, x, y) that decides if the point (x,y) is contained in the range of array img. (within.rb)
   HINT: We can use within_range(a,i) defined as in the last question.
5. Define the following functions. Note that it is better to put the functions in a file whose name is given between ( ).
   (a) a function solutions(a,b,c) that computes *the number of* real solutions in the quadratic equation $ax^2 + bx + c = 0$. Consider not only the discriminant but also the case when $a = 0$ or $b = 0$. (quadratic.rb)
   (b) a function solve1(a,b,c) that computes one of real solutions in the quadratic equation $ax^2 + bx + c = 0$ if it has. If it has no solution, then return "nil" (nothing). Consider not only the discriminant but also the case when $a = 0$ or $b = 0$. (quadratic.rb)

(c) a function median(x,y,z) that computes the median of them, where the median is the middle value if we sort them in nonincreasing order. (median.rb)

(d) a function income_tax(income) that computes the income tax in Japan when you earn income thousand yen. Suppose that the rate of income tax is determined at 2010 as follows. (tax.rb)

- 5% if the income is at most 1950 thousand yen.
- 10% if the income is more than 1950 and at most 3300 thousand yen.
- 20% if the income is more than 3300 and at most 6950 thousand yen.
- 23% if the income is more than 6950 and at most 9000 thousand yen.
- 33% if the income is more than 9000 and at most 18000 thousand yen.
- 40% if the income is more than 180000 thousand yen.

(e) a function days_of_february(year) that computes the number of days in February in year. (calendar.rb)

(f) a function days_of_month(year, month) that computes the number of days of month in year. (calendar.rb)

6. A *logic function* is a function such that the parameters are given by TRUE/FALSE and the returned value is also TRUE/FALSE. We often use 1/0 instead of TRUE/FALSE with the same meaning. Examples of a logic function are && and ||, whose returned values are as follows.

x	y	x && y	x || y
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Define the logic functions xor(x,y) and implies(x,y) whose returned values are given as in the following table.

x	y	xor(x,y)	implies(x,y)
False	False	False	True
False	True	True	True
True	False	True	False
True	True	False	True

7. Make a function that decides whether given three numbers a, b, c form the sides of a triangle.

8. The function `rand()` is implemented in Ruby, and it returns a random number between 0 and 1. Make a function `double(x)` that returns the double of the given number `x` with probability 1/2, and `x` otherwise. That is, `x` doubles if `rand()` is at least 0.5, and remains unchanged otherwise.

9. The function `rand()` is implemented in Ruby, and it returns a random number between 0 and 1. Make a function `average(n)` that returns the average value when we call `rand()` `n` times.

10. (a) Make a function `print_divnum(n)` that prints all the numbers between 1 and a given number n that are divisible by 7, but not divisible by 3. Use `print` to display numbers, and the operator `&&`.

    (b) Make a function `print_divnum2(n)` that enumerates all the numbers between 1 and a given number n that are divisible by 7 or divisible by 11. Use the operater ||.

4 Arrays and Images

1. Make the following array, and display it with the function show on isrb.

```
w=[
[0,1,1,1,1,1],
[0,1,0,0,0,1],
[0,1,0,1,0,1],
[0,1,1,1,0,1],
[0,1,0,0,0,1],
]
```

2. We can draw a color image using the function show. Below is an example of a $2 \times 3 \times 3$ array.

```
d=[[[0,0,0],[0,1,0],[0,0,1]],
[[1,0,0],[1,1,0],[1,0,1]],
[[0,0,0],[0,1,0],[0,0,1]],
[[1,0,0],[1,1,0],[1,0,1]]]
```

   Perform show(d) on isrb. Each entry of d has three values, which represent Red, Green, and Blue, respectively.

3. Using show, draw simple national flags with colored image. For example,

Green	White	Red

Table 1: Italy

Green	White	Orange

Table 2: Ireland

Red
White
Blue

Table 3: Italy

   Note that we can use the following table

Color	Green	White	Red	Orange	Blue
Red	0	1	1	1	0
Green	0.6	1	0	0.4	0.2
Blue	0	1	0	0	0.6

4. Solve the following.(You can refer to the sources on ITC-LMS)
   (a) Define a function make1d(n) that makes a 1-dimensional array with size n each of whose entry is 0.
   (b) Define a function make2d(h,w) that makes a 2-dimensional array with h rows and w columns each of whose entry is 0.

(c) Define a function make2d_color(h,w) that makes an h×w×3 array each of whose entry is 0.

5. Define a function gradation(n) in gradation.rb that makes a one-dimensional array with size n whose ith value is equal to i/n.

   Hint: Change some lines in make1d so that we can assign a value using i in each iteration.

6. Let $\mathrm{i} = \sqrt{-1}$. We represent a complex number $x + y\mathrm{i}$ as an array of length two. For example, a=[1,2] means $1 + 2\mathrm{i}$.

   (a) Define a function add(a,b) that returns the sum of two complex numbers a and b. The returned value is also an array of length two.

   (b) Define a function mult(a,b) that returns the product of two complex numbers a and b.

   (c) Define a function abs(a) that returns the absolute value of a complex number a.

   (d) Define a function div(a) that returns the result when we divide a by b.

7. Let a be a one-dimensional array with size n. We regard it as a vector of order n.

   (a) Define a function vec_mult(a,b) that returns the (inner) product $a^\top b$ of a and b.

   (b) Define a function vec_norm(a) that returns the norm of a, that is, $\sqrt{a^\top a}$.

8. Let a and b be two two-dimensional arrays with size $n \times n$. We consider a and b as matrices. You can use the functions in the last problem.

   (a) Define a function mat_add(a,b) that returns the sum of a and b.

   (b) Define a function mat_mult(a,b) that returns the product of a and b.

9. (a) Make a function length3(a,x) that computes the number of elements around the xth element in a, that is,

$$\mathsf{length3}(\mathsf{a},\mathsf{x}) = \begin{cases} 3 & \text{if } 0 < x < \ell - 1 \\ 1 & \text{if } x = 0 \text{ and } \ell = 1 \\ 2 & \text{if } \ell > 1 \text{ and } (x = 0 \text{ or } \ell - 1) \\ 0 & \text{otherwise} \end{cases}$$

   where ℓ is the length of the array a. Note that the indices of a start from 0 to $\ell - 1$.

   (b) Make a function array_average3(a,x) that computes the average of the elements around the xth element in a. For example, array_average3(a,2) is equal to (a[1]+a[2]+a[3])/3 if the length of a is at least 4, and array_average3(a,0) is equal to (a[0]+a[1])/2.

(c) Make image_average9(image,x,y) that computes the average of the elements around the xth element in a 2-dimensional array image. For example, image_average3(a,0,0) is equal to (a[0][0]+a[0][1]+a[1][0]+a[1][1])/4, and if x,y are not on the "boundaries", the value is the average of the 9 elements around a[x][y] with a[x][y] itself.

10. We can make an image by modifying another image. For example, we can make a gray image(array) s brighter by replacing each entry b with $(b+1)/2$. Thus, it is realized by making a new array img with the same size as s, and defining img[y][x]=(s[y][x]+1)*0.5 as the brightness of the (x,y) coordinate.

    Define the following functions and apply them to your image.

    (a) a function brighter(img) that makes a given image img brighter.

    (b) a function blend(img1, img2) that blends two images img1 and img2, where "blend" means to take the average of the two values of img1 and img2 at the same coordinates.

    (c) a function blur(img) that "average" the image img using the function image_average9 in the previous problem.

11. Compare the outputs of the following two programs. (See also Appendix in lecture slides)

```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```

```
b = Array.new(2)
for i in 0..1
   b[i] = Array.new(2)
   for j in 0..1
        b[i][j] = i
    end
end
b
```

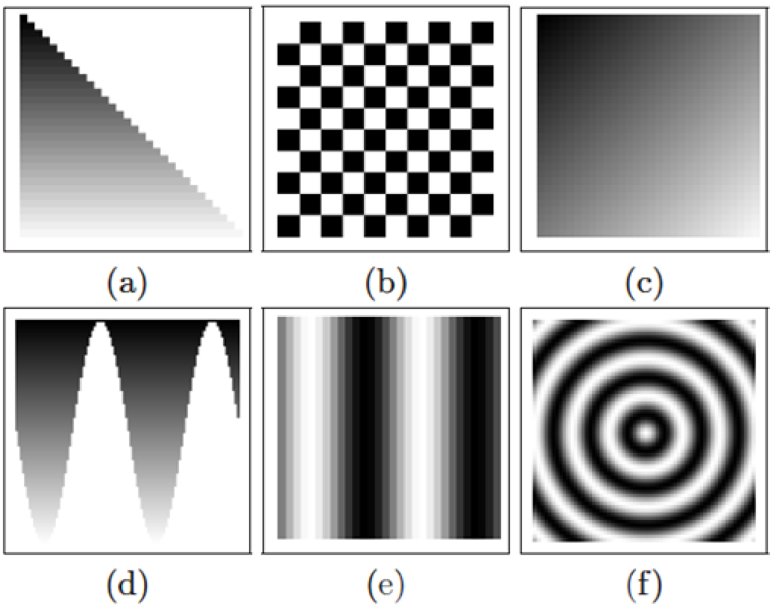
12. Compare the outputs of the following two functions. (See also Appendix in lecture slides)

```
def inc1(b)
  n = b.length()

  for i in 0..n-1
        b[i] = b[i]+1
  end
  b
end
```

```
def plus1(b)
   n = b.length()
   c = Array.new(n)
   for i in 0..n-1
      c[i] = b[i]+1
   end
   c
end
```

13. Make functions that generate the following images.


(a)
(b)
(c)
(d)
(e)
(f)

5 Iteration and Recursion

1. **(Review)** Consider computing the summation from 1 to n.

   (a) Confirm that the following program using for works.

   ```
   def sum_loop(n)
     s = 0
     print "sum=", s, "\n"
     for i in 1..n do
        s = s + i
        print "sum=", s, "\n"
     end
     s
   end
   ```

   (b) Confirm that the following program using recursion works.

   ```
   def sum(n)
     print "Compute sum(",n,")...\n"
     if n >= 2
       print "sum(", n, ")=sum(", n-1 ,")+", n, "\n"
       s = sum(n-1) + n
     else
       s = 1
     end
     print "sum(", n, ")=", s, "\n"
     s
   end
   ```

2. Consider the function mult_sum(p,n) that computes the sum of multiples of p between 1 and n ($p \leq n$).

   (a) Make the function using repetition.

   (b) Make the function using recursion.

3. **(Using repetition)**. Define the following functions using repetition.

   (a) a function factorial_loop(n) that computes the factorial of n, that is, the product of all positive integers less than or equal to n. (factorial_loop.rb)

   (b) a function power2_loop(n) that computes 2^n. Do not use **. (power_loop.rb)

   (c) a function power_loop(x, n) that computes x^n. Do not use **. (power_loop.rb)

   (d) a function taylor_e_loop(x, n) that computes the following series

   $$\sum_{k=0}^{n} \frac{\mathsf{x}^k}{k!}.$$

   Note that this is the Taylor series of e^x when $n \to \infty$. (taylor_e_loop.rb)

4. **(Using recursion)**. Define the following functions using recursion.

   (a) a function factorial(n) that computes the factorial of n, that is, the product of all positive integers less than or equal to n. (factorial.rb)

   (b) a function power2(n) that computes 2^n. Do not use **. (power.rb)
   Hint: 2^n is equal to $2 \times 2^{n-1}$.

   (c) a function power(x, n) that computes x^n. Do not use **. (power.rb)

   (d) a function taylor_e(x, n) that computes the following series

   $$\sum_{k=0}^{n} \frac{\mathsf{x}^k}{k!}.$$

   Note that this is the Taylor series of e^x when $n \to \infty$. (taylor_e_loop.rb)

5. **(Using repetition)**. Define the following functions using repetition. (prime_loop.rb)

   (a) a function nod_loop(k,n) that computes the number of divisors of k among all positive integers less than or equal to n.

   (b) a function nop_loop(n) that computes the number of prime numbers among all positive integers less than or equal to n.

   (c) a function msod_loop(n) that computes the maximum sum of divisors, that is, the integer k in 1,...,n such that the sum of divisors sod(k,k) is maximized.

6. **(Using recursion)**. Define the following functions using recursion. (prime.rb)

   (a) a function nod(k,n) that computes the number of divisors of k among all positive integers less than or equal to n.

   (b) a function nop(n) that computes the number of prime numbers among all positive integers less than or equal to n.

   (c) a function msod(n) that computes the maximum sum of divisors.
   Hint: Let sod(k,k)=s_k. Then this problem is equivalent to finding the maximum of $s_1, \ldots, s_n$. The maximum of $s_1, \ldots, s_n$ is equal to "the maximum of the maximum of $s_1, \ldots, s_{n-1}$" and s_n. That is, we have the following relation.

   $$\mathsf{msod}(n) = \begin{cases} \mathsf{msod}(n-1) & \text{if } \mathsf{msod}(n-1) \geq s_n \\ s_n & \text{if } \mathsf{msod}(n-1) < s_n \\ s_1 & \text{if } n = 1. \end{cases}$$

7. **(Using repetition)**. Define the following functions using repetition.

   (a) Define a function np_loop(n) that computes the next prime number (the minimum prime greater than or equal to n). If n is a prime, then np_loop(n)=n. (prime_loop.rb)

   (b) Define a function nth_prime_loop(p,n) that computes the nth prime number greater than the prime p. For example, nth_prime_loop(5,3) is 13. (prime_loop.rb)

(c) Define a function tnpo(n) that returns the half of n if n is even, and 3n+1 if n is odd. Mathematician Collatz conjectured[1] that every integer n comes to 1 by applying tnpo repeatedly. For example, if we have 3, then we obtain $3 \Rightarrow 10 \Rightarrow 5 \Rightarrow 16 \Rightarrow 8 \Rightarrow 4 \Rightarrow 2 \Rightarrow 1$ by applying tnpo repeatedly.

Here, we define a function collatz (n) that computes the number of repetition times that we apply tnpo to come to 1. For example, collatz(16)=4, collatz(5)=5, and collatz(3)=7. Using the repetition, make a Ruby function collatz_loop (n) to compute collatz (n). (collatz_loop.rb)

8. **(Using recursion)**. Define the following functions using recursion not using repetition.

   (a) Define a function np(n) that computes the next prime number (the minimum prime greater than or equal to n). If n is a prime, then np(n)=n, and if n is not a prime, then it is equal to the minimum prime which is at least n+1. (prime.rb)

   (b) Define a function nth_prime(p,n) that computes the nth prime number greater than the prime p. For example, nth_prime(5,3) is 13. In fact, nth_prime(5,3) is equal to nth_prime(7,2), because the next prime is 7. Since the next prime is 11, it is equal to nth_prime(11,1), and therefore it is equal to the next prime 13. (prime.rb)

   (c) Make a function collatz (n) defined in the previous problem.
   Hint: Consider the relationship between collatz (n) and collatz (tnpo(n)). (collatz.rb)

9. The operation to concatenate two arrays is represented by +. For example, [0]+[0]=[0,0] and [1,2]+[3,4]=[1,2,3,4].

   (a) Re-define the function make1d(s) in a recursive way.

   (b) Re-define the function make2d(h,w) in a recursive way.

10. **(Making an array)** Define the following functions to make an array with dimension at least three.

    (a) Define a function make3d(a,b,c) that makes an a×b×c array all of whose entries are 0, based on the definition of make2d(a,b,c). (make3d.rb)

    (b) Define a function makend(n,m) that makes a m× ⋯ ×m(n times) array. (makend.rb)
    Hint: Consider a recursive definition.

    (c) Define a function makearray(d) that makes an array whose size is defined by an array d. For example, makearray([2,4,3]) makes a $2 \times 4 \times 3$ array. (makearray.rb)

[1]this problem is an open problem in mathematics, that is, there are no proof and no counterexample.

5.1 Applications

11. **(combination number by recursion)** The combination number ${}_nC_k$ is the number of combinations of choosing k items out of n items.

    (a) Using the fact
    $$ {}_nC_k = \frac{n!}{k!(n-k)!}, $$
    explain why it holds that
    $$ {}_nC_k = \begin{cases} 0 & \text{if } k > n \\ 1 & \text{if } k = 0 \\ {}_{n-1}C_{k-1} + {}_{n-1}C_k & \text{otherwise.} \end{cases} $$

    (b) Define a function combination(n,k) that computes the combination number in a recurive way.

12. **(combination number by repetition)** Using FOR-loop, make a function combination_loop(n,k) that computes ${}_nC_k$. Compare combination(n,k) and combination_loop(n,k) increasing n and k, and discuss which is faster.

13. **(Sierpinski Triangle)**. Define a function sierpinski(n) that makes a 2-dimensional array with size n× n such that the (i, j) entry is equal to the remainder of ${}_iC_j$ when divided by 2. (if $i < j$ then the value is 0) Use the function show to display the obtained array in isrb. The results will be depicted as in Figure 1, where black and white are reversed for visibility.

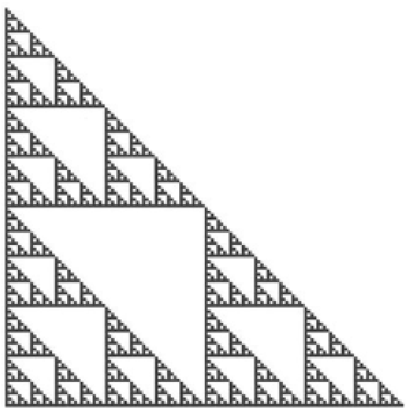


Figure 1: Sierpinski Triangle

14. **(Tower of Hanoi)**. The goal of the game "Tower of Hanoi" is to move all the disks from the left peg to the middle one. Only one disk may be moved at a time. A disk can be placed either on an empty peg or on top of a larger disk. Try to move all the disks using the smallest number of moves possible.

(a) Let Hanoi(n, a, b, c) be a function that describes a procedure to move n disks from the left beg a to the middle peg b using the right peg c. Complete the program in Slides.

(b) Let Hanoi_times(n, a, b, c) be a function computes the minimum number of times of moving disks when we do Hanoi(n, a, b, c). Complete the program in Slides.

(c) Compute Hanoi(4,"a","b","c"), Hanoi(5,"a","b","c"), Hanoi(6,"a","b","c"), and Hanoi(50,"a","b","c").

15. Make a program that displays the following arrow, in which we can determine the size(height) of the arrow.

```
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************
         *
         *
         *
         *
         *
         *
         *
         *
         *
         *
```

16. We want to define a function that converts a number n to binary. Make such a function in two ways: using repetition and recursion.

17. (a) Consider finding the maximum value in a given array a. Define such a function maxElem(a,i) using recursion, where it returns the maximum value of a[0],..., a[i]. If i=0, return a[0]. Otherwise, if `maxElem(a,i-1) < a[i]` then return a[i], and otherwise return maxElem(a,i-1). That is, the maximum value of the array a can be obtained by maxElem(a,n-1).

(Example: maxElem([4,3,5,1],3)=5.)

(b) Consider finding the second largest value of a given array a. Define a function secondMaxElem(a) using recursion.

Hint: Let max be the maximum value between the first i-1 elements in a, and max2 be the second largest one. Define secondMaxElem2(a,max,max2,i) that returns the second largest value in a[0],…, a[i]. This can be used to define secondMaxElem(a). (We may suppose that all the entries in the array a are distinct.)

6 Algorithms and Complexity

6.1 Fibonacci Numbers

1. **(Same as in Slides)**Is the number of the 100th Fibonacci number odd or even? Explain why.

2. **(Same as in Slides)** Make the programs fibr(k) and fibl(k). Confirm that they return the same values for some k's. Make a program that decides if fibr(k) and fibl(k) are same for k=1,...,p with a given p.

3. Let $\phi = \frac{1+\sqrt{5}}{2}$. It is known (see also Problem 5 below) that fib(k) is approximated by

$$\mathsf{fib(k)} \approx \frac{\phi^k}{\sqrt{5}}.$$

   Define a function fiba(k) that computes the right-hand side of the equation, and examine the difference between fiba(k) and fibr(k) for k=1,...,p with a given p.

4. **(Same as in Slides)** Make the functions fibr(k) and fibl6(k), and record the computational times t_r and t_l for the two functions using run. Estimate A, B, C that satisfy $t_r(k) \simeq A \cdot B^k$ and $t_l(k) \simeq Ck$.

5. Explain why fibr(k) takes $4f(k) - 3$ operations. Hint: we can use either of the following strategies.

   - Show $T_r(k) = 4f(k) - 3$ if $k \geq 1$ by induction using $T_r(k) = T_r(k-1) + T_r(k-2) - 3$ and $T_r(0) = T_r(1) = 1$, or
   - Estimate the number of nodes and edges in the generated tree as in Slides.

6. Let f_k be the kth Fibonacci number. Then it holds that

$$\begin{pmatrix} f_{k+1} \\ f_k \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_k \\ f_{k-1} \end{pmatrix}.$$

   (a) Explain why the above equation holds.

   (b) Let A be the matrix in the form of

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

   Determine the eigenvalues λ_1 and λ_2 of A.

   (c) By the definition of eigenvalues, there exists a nonsingular(invertible) matrix U such that

$$A = U \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} U^{-1}$$

   Determine the matrix U and U^{-1}.

(d) By (a), we have

$$\begin{pmatrix} f_{k+1} \\ f_k \end{pmatrix} = A^k \begin{pmatrix} f_1 \\ f_0 \end{pmatrix} = A^k \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Using the equation in (c), express the kth Fibonacci number f_k with λ_1 and λ_2.

7. Define the function matpower(a, n) that computes the nth power of a matrix a. We may suppose that a is a 2×2 matrix and it is given as a 2-dimensional array.

(a) Define the function matmul(a,b) that computes the product of two matrices a and b.

Recall: When two matrices have size 2 by 2, it holds that

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} = \begin{pmatrix} a_{00}b_{00} + a_{01}b_{10} & a_{00}b_{01} + a_{01}b_{11} \\ a_{10}b_{00} + a_{11}b_{10} & a_{10}b_{01} + a_{11}b_{11} \end{pmatrix}.$$

(b) Define the function matsquare(a) that computes the square of a matrix a.

(c) Define the function matpower_loop(a,n) that computes the nth power of a matrix a using for-loop.

(d) We can observe that the power of matrices can be computed more efficiently. For example, for a matrix A, $A^{16} = (((A^2)^2)^2)^2$. Hence A^{16} can be obtained by taking a square four times. This reduces the number of matrix multiplications from 16 to 4. More precisely, we have the following.

$$A^n = \begin{cases} I & \text{if } n = 0, \\ (A^{n/2})^2 & \text{if } n \text{ is an even number with } n \geq 2, \\ A \times A^{n-1} & \text{if } n \text{ is an odd number,} \end{cases}$$

where I is the identity matrix (unit matrix). For example, if we have A^{20}, then we have

$$\begin{aligned} &A^{20} \Rightarrow (A^{10})^2 \Rightarrow ((A^5)^2)^2 \Rightarrow ((A \times A^4)^2)^2 \Rightarrow ((A \times (A^2)^2)^2)^2 \\ &\Rightarrow ((A \times ((A^1)^2)^2)^2)^2 \Rightarrow ((A \times ((A^1 \times A^0)^2)^2)^2)^2 \Rightarrow ((A \times ((A^1 \times I)^2)^2)^2)^2. \end{aligned}$$

Thus the number of matrix multiplications to obtain A^{20} is 6.

Using this relationship, make a recursive program matpower(a,n) that computes the nth power of a matrix a.

(e) Estimate the computational time complexity of matpower(a,n) using the order notation.

8. Define the function fibm(k) that computes the kth Fibonacci number using matpower(a,n) in the previous exercises. Furthermore, confirm that it returns the same values as fibl and fibr.

9. Define the function fibm6(k) that computes the first 6 digits of the kth Fibonacci number using matpower(a,n). Using bench.rb, confirm that the computational time is proportional to $\log k$, which means $O(\log k)$.

6.2 Computational Complexity and Sorting

1. **(Combination number, same in slides)** Estimate computational complexities of the two programs to compute the combination numbers in Exercises 5.1.11–12 using recursion and repetition, respectively.

2. A certain store has two software A and B to process experimental data. It is known that A can process in $O(N^2)$ time, while B can process in $O(N \log_2 N)$ time, when the data size is N. For 1000-record test data, Software A takes 1 second, while Software B takes 10 seconds. The target data has 1-million records. Which software is better to process the target data?

3. **(Simple Sort, same in slides)** Define a function min_index(a,i) that returns the index of the minimum value in a[i],...,a[n-1], where n is the length of a. Complete the program of the simple sort algorithm using this function.

4. **(Merge Sort, same in slides)** Make the missing parts in the program in Slides, and complete the function merge(a,b). Confirm that it works by executing merge([3,5,9], [1,4,6,7,8]) and merge([0,0.5,1.0], [0,0.9,1.0]).

5. Consider to compare the computational times of simple sort and merge sort. In the merge sort algorithm, we need to do some complicated tasks such as "making a new array." Hence it is perhaps expected that the merge sort algorithm has more time than the simple sort algorithm.

   (a) Make a randomly-generated sequence. This can be done by executing randoms(id, size, max) in a downloadable program randoms.rb.

   Remark: randoms(id, size, max) returns an array with size size, where each entry is a random value which is at least 0 and less than max. Note that if max is 1, it returns a real, and, if max is a positive integer larger than 1, it returns an integer. The integer(parameter) id means an index, and if id, max, size are all same, it returns the same array.

   (b) Use run(f, x, y) in bench.rb in a similar way to the Fibonacci case to measure the computational times.

   (c) Define the function compare_sort(max, step) to compare two sorting algorithms. This function first makes random arrays whose sizes are step, 2step, 3step, ..., max, and measure computational times for the simple sort and the merge sort.

```
load("./randoms.rb") # randoms(id, size, max)
load("./bench.rb") # run(function_na,e, r, v)
load("./simplesort.rb") # simplesort(a)
load("./mergesort.rb") # mergesort(a)

def compare_sort(max, step)
   for i in 1..(max/step)
      x = i*step
      a = random(i,x,1)
      run("simplesort", x, a)
```

```
            a = random(i,x,1)
            run("mergesort", x, a)
        end
    end
```

Using the function compare_sort(max,step), discuss the differences between two algorithms

6. **(Recursive Definition of Merge Sort)** By the following instruction, define the function mergesort_r(a) that executes the merge sort algorithm recursively.

   Let us first consider the final step of the merge sort algorithm. At this step, we have two sorted sequences p and q, where p is obtained by sorting the first half of a, and q is obtained from the latter half of a. We can use the function merge, which has already defined, to merge the two sorted sequences p and q.

   To obtain p and q, it suffices to do the merge sort recursively. That is, we apply the merge sort to a part of the array a. Thus we introduce the function doing "sort the 1st to rth elements in a".

   Let us consider defining the function merge_rec(a, l, r) that sorts a[l],...,a[r].

   - If l=r, then it returns an array with size 1 consisting of only a[l]. (We need not to sort)
   - If l<r, then we divide the array a into two parts, apply recursively the merge sort, and merge them. That is,
     (a) Define m=(l+r)/2.
     (b) Apply merge_rec to the elements from lth to mth and the elements from (m+1)th to rth, respectively. Denote the obtained sorted sequences by b and c.
     (c) Return the sequence obtained by executing merge to b and c.

   When we can define merge_rec(a, l, r), mergesort_r(a) can be defined by only calling merge_rec(a,0,a.length()-1).

6.3 Problems from Past Exams

1. **(Past Exam 2010)** Suppose that an array a has size n and contains m kinds of positive integers. We want to store all the distinct integers of a to another array b of size m, and also return the frequencies of occurrence in an array c of size m. For example, if a=[3,1,4,1,5,9,2,6,5,3], then n is 10 and m is 7. In this case, b contains [3,1,4,5,9,2,6], and c contains [2,2,1,2,1,1,1], which means that a has two 3's, two 1's, and so on.

   (a) The following program is a program to compute b and c from a. Describe the computational complexity using n and m. Note that the parameters b and c are supposed to be arrays of size m.We suppose that each entry in array b is initialized to be 0.

```
def intcount(a, b, c)
   for i in 0..(a.length()-1)
      x = a[i]
      j = 0
      while b[j] != 0 && b[j] != x
         j = j + 1
      end
      if b[j] == 0
         b[j] = x
         c[j] = 1
      else
         c[j] = c[j] + 1
      end
   end
end
```

   (b) Suppose that a is sorted, that is, elements in a are ordered in nondecreasing order. Modifying the above program, make a new function intcount(a,b,c) that runs in $O(n)$ time.

2. **(Past Exam 2011)** Let a be an array of N integers, denoted by a $= [x_0, x_1, \ldots, x_{N-1}]$. Let s(a, i, j) be the sum of the integers from a[i] to a[j-1]$(0 \leq i \leq j \leq N)$ (when $i = j$, define s(a, i, j)$= 0$). Answer the following questions.

   (a) When a = [8, -4, -5, 2, 4, -5, 5, 3, -7, 8], we have s(a, 0, 0)$= 0$ and s(a, 0, 1)$= 8$. Calculate s(a, 0, 2), s(a, 0, 3), and s(a, 0, 4).

   (b) Using N, describe the computational complexity (order) of an algorithm using simple iteration to compute s(a, 0, N).

   (c) Let mss(a, x, y) be the maximum value of s(a, i, j) for all i, j with x $\leq$ i $\leq$ j $\leq$ y (we suppose 0 $\leq$ x $\leq$ y $\leq$ N). We make a program mss(a, 0, N) as below by computing s(a, i, j) for all pairs i and j. Describe the computational complexity of mss(a, 0, N) using N.

```
def mss(a,x,y)
  m = 0
  for i in x..y
    for j in i..y
       m = max(m, s(a,i,j))
    end
  end
  m
end
```

(d) It is known that if mss(a, 0, z - 1) = s(a, x, z - 1) and s(a, x, z) < 0, then a[z-1] is not contained in the interval that gives mss(a, 0, y) (in other words, mss(a, 0, y) = s(a, i, j) then z does not satisfy i $\leq$ z $\leq$ j). Using this fact, we can make the following program mss0(a,m) to compute mss(a, 0, m).

```
def mss0(a,m)
  t = 0
  sum = 0
  for i in 0..(m-1)
    sum = sum + a[i]
    if sum > t
      t = sum
    else
      if sum < 0
        sum = 0
      end
    end
    # (*)
  end
  t
end
```

When we apply mss0(a, 10) with a=[8,-4,-5,2,4,-5,5,3,-7,8], calculate the values of sum and t at the end (∗) of each repetition using the following table.

i	0	1	2	3	4	5	6	7	8	9
sum										
t										

(e) Describe the complexity order of mss0(a,N) discussed in (d), using N.

3. **(Past Exam 2011)** Reading the Ruby program, answer the following.

```
def f(x,y)
  z = 1
  while y != 0
    while y % 2 == 0
      x = x * x
      y = y / 2
```

```
    end
    y = y - 1
    z = z * x
  end
  z
end
```

(a) Present the results when we call this function with parameters f(2,4) and f(3,5).

(b) Let c = f(a,b). Explain concisely the relationship between c and a, b. Note that a and b are supposed to be nonnegative integers.

4. **(Past Exam 2011)** Suppose that there is a point whose x-coordinate is nonnegative. The point moves in one step through linear distance 1 toward randomly-chosen direction. If the point bumps into the y-axis, it will be reflected completely. In this case, the moving distance is the sum of distances before and after the reflection, which has to be one. However, we have a hole from $(0, -1)$ to $(0, 1)$.

   Let (x, y) be the initial coordinates of the point. We describe a procedure escapesteps(x,y) to compute the number of steps until the point passes through the hole.

```
include(Math)

def escapesteps(x,y)
  n = 0
  escaped = false
  while !escaped
    r = rand()
    dx = cos(2*3.14159265358979*r)
    dy = sin(2*3.14159265358979*r)
    x1 = x + dx
    y1 = y + dy
    if x1 < 0
      if (A)
        escaped = true
      else
        (B)
        y = y1
      end
    else
      (C)
      y = y1
    end
    n = n + 1
  end
  n
end
```

Note that the function rand() returns a (pseudo)random real number between 0 and 1.

(a) (A) represents a condition that the point passes through the hole. Fill in (A).

(b) (B) and (C) update the x-coordinate. Fill in (B) and (C).

5. **(Past Exam 2012)**

(a) Explain what f(5) returns.

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

(b) The above function f is written using recursion. Without using recursion, redefine f using only "while" "if" "for".

(c) Explain what g(5) computes.

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

(d) The function h is the function that we rewrite the above function g without recursion. Fill in the blanks.

```
def h(n)
  result = make1d(n+1)
  i = (A)
  while (B)
    if i >= 2
      result[i] = (C)
    else
      result[i] = 1
    end
    i = (D)
  end
  result[ (E) ]
end
```

6. **(Past Exam 2012)** Suppose that an array `a` contains `n` integers (`n` $\geq$ 2). Consider computing the minimum of absolute values of differences between two entries in `a`. For example, if `a` = [9, 3, 5], the answer is 2.

   (a) Describe an algorithm whose complexity order is $O(n^2)$. You can answer using sentences or as a Ruby program.

   (b) Assume that elements in `a` are sorted in nondecreasing order. Describe an algorithm that runs in $O(n)$ time under the assumption.

   (c) Consider an algorithm whose complexity order is better than one in (a), not assuming that `a` is sorted. Explain the complexity order with some reason.

7. **(Past Exam 2013, PEAK)** Consider an algorithm to compute the greatest common divisor of two integers `a` and `b` (`a<b`). The greatest common divisor (gcd, for short) of `a` and `b` is defined to be the largest positive integer that divides the numbers `a` and `b` without a remainder. For example, the gcd of 12 and 18 is 6.

   (a) Since the greatest common divisor is between `1` and `a`, we can make the following function `gcd_loop` using repetition. Fill in the blanks $\boxed{\text{(i)}}$ to $\boxed{\text{(iii)}}$.

```
def gcd_loop(a,b)
  result = 0
  for i in 1..a
    if [(i)] && [(ii)]
      [(iii)]
    end
  end
  result
end
```

   (b) We denote by `gcd(a,b)` the greatest common divisor of `a` and `b`. Letting `r` be the remainder when `b` is divided by `a`, we have `gcd(a,b)=gcd(r,a)`. Explain why this equation holds.

   (c) Based on the relationship described in (b), we can make the following recursive program `gcd_r(a,b)`. Fill in the blanks. Note that each box may have multiple lines.

```
def gcd_r(a,b)
  if a == 0
  [(iv)]
  else
  [(v)]
  end
end
```

   (d) Suppose that `a=273` and `b=504`. How many times do we call `gcd_r` in `gcd_r(a,b)`?

(e) We would like to discuss the computational complexity of the function `gcd_r(a,b)`. First, explain why we always have `b` $\geq$ `2r` for the remainder `r` of `b` when divided by `a`. Then describe how many times we need to call `gcd_r(a,b)` using `a` and `b`.

7 Dynamic Programming

1. **(Past Exam 2010)** There is a (M+1)×(N+1) grid. Suppose that a piece moves from the bottom-left square with coordinate $(0,0)$ to the top-right square with coordinate (M,N). The piece can move one step forward either "right", "up", or "upper-right" at one time. Answer the following questions.

   (a) Let $T_{m,n}$ be the number of possible routes from coordinate $(0,0)$ to (m,n). $(1 \le m \le M,\ 1 \le n \le N)$. Then explain we have

   $$T_{m,n} = T_{m,n-1} + T_{m-1,n} + T_{m-1,n-1}$$

   Moreover, describe how to obtain the initial terms

   $$T_{0,0}, T_{0,i}, T_{j,0} (1 \le i \le M, 1 \le j \le N)$$

   (b) Explain the overview of an algorithm to compute $T_{M,N}$ using dynamic programming.

2. **(Revision of Past Exam 2014, PEAK)** Suppose that the Bank of Japan decided to use only three kinds of coins, 1 yen, 4 yen, and 5 yen. Consider the situation when we pay m yen using such coins. For example, we can pay $m = 7$ yen in one 4-yen coin and three 1-yen coins. We would like to minimize the number of coins used for a given integer m. If $m = 7$, the minimum number is three when we use one 5-yen coin and two 1-yen coins.

   (a) We first consider the following greedy strategy to reduce the number of coins: repeatedly use the largest coin which we can use. Fill in the blanks.

```
def greedy(m)
   num = 0
   while m > 0
     if m >= 5
       m = m - [(i)]
     elsif m >= 4
       m = m - [(ii)]
     else
       m = m - [(iii)]
     end
     num = [(iv)]
   end
   num
end
```

   (b) We found that the function greedy does not return the minimum number. Give an example of m that the above program does not work correctly, and compare the output of greedy(m) and the minimum number.

We next consider applying a technique called dynamic programming.

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f(m)$	0															

(a) Let $f(m)$ be the minimum number of coins to pay m yen. Fill in the table ($m \leq 15$).

(b) Find a recursive relationship for the values $f(m)$. Here, $\min\{a_1, \ldots, a_p\}$ means the minimum value of the numbers $a_1, \ldots, a_p$.

$$f(m) = \begin{cases} 0 & \text{if } m = 0 \\ f(m - \boxed{\text{(v)}}) + 1 & \text{if } 0 < m < 4, \\ \min\{f(m - \boxed{\text{(v)}}) + 1, f(m - \boxed{\text{(vi)}}) + 1\} & \text{if } 4 \leq m < 5, \\ \min\{f(m - \boxed{\text{(v)}}) + 1, f(m - \boxed{\text{(vi)}}) + 1, f(m - \boxed{\text{(vii)}}) + 1\} & \text{otherwise.} \end{cases}$$

(c) Using the above relationship, make a Ruby function based on dynamic programming by filling in the blanks.

```
def pay(m)
  coins = [1,4,5] # the types of coins
  f = make1d(m+1)
  f[0] = (viii)
  for j in 1..m
    for i in 0..2
      if m - coins[i] >= 0
        if (ix) < (x)
          f[j] = (ix)
        end
      end
    end
  end
  (xi)
end
```

(d) Estimate the computational complexity of pay(m) using m.