Rプログラミング（演習解答）

基本問題（解答）

〇関数定義

問題１（1）：あるライブコンサートのチケットの料金は次の通りである。

S席：10000円，A席：8000円，B席：5000円

このとき，各チケットの枚数を変数とし，出力を売上額とする関数riekiを定義しなさい.

解答例：

```
rieki  <-  function(a,b,c){

     x <- 10000*a+8000*b+5000*c

     return(x)
}
```

```
reiki(2,3,4) = 64000
```

基本問題（解答）

〇関数定義

（２）（ヘロンの公式）３つの正の数値(a, b, c)を入力して，その長さを三辺とする三角形の面積を求める関数 heron01を作成しなさい.

出力例：
heron01(2,2,2)=1.732051

（解答例）

```
Heron01 <- function(a,b,c){
      s = (a+b+c)/2
      v = sqrt(s*(s-a)*(s-b)*(s-c))
  return(v)
}
```

基本問題

○if 文

問題2（1）　AとBが買い物に行くとする．Aがある書籍を購入したい。

もしもAの所持金で足りればそのまま支払う。
もしもAの所持金で足りない場合，Bから借りて支払う。
もしもAの所持金で足りない場合，Bから借りても支払うことができなければ購入しない。

A, Bの所持金と書籍の値段を入力したとき，Bからの借入金額を出力する関数loanを定義しなさい。

出力例：

```
loan(200,300,150) = 0
loan(200,300,400) = 200
loan(200,300,900) = ”too expensive”
```

基本問題（解答）

問題2（1）（解答）

```
loan <- function(a,b,c){
     x<-0
     if(a>=c){
        x <- 0
     }
     else if(a+b >= c){
        x <- c-a
     }
    else{
         x <- "too expensive"
    }
    return(x)
}
```

出力例：

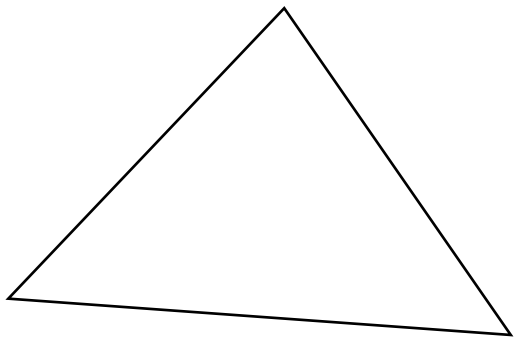
```
loan(200,300,150)= 0
loan(200,300,400)= 200
loan(200,300,900)= ”too expensive”
```

基本問題

〇 if文（ヘロンの公式）

（2） 3 つの正の数値(a, b, c)を小さい順に入力して，その長さを三辺とする三角形の面積を求める関数を heron02 作成しなさい。ただし，3 つの数値から三角形が出来ない場合は999を出力する.

（3） 3 つの正の数値(a, b, c)をランダムに入力して，その長さを三辺とする三角形の面積を求める関数 heron03 を作成しなさい。ただし，3 つの数値から三角形が出来ない場合は999を出力する.



基本問題

○ if文（ヘロンの公式）（解答）

（2）３つの正の数値(a, b, c)を小さい順に入力して，その長さを三辺とする三角形の面積を求める関数を heron02 作成しなさい。ただし，３つの数値から三角形が出来ない場合は999を出力する.
（3）３つの正の数値(a, b, c)をランダムに入力して，その長さを三辺とする三角形の面積を求める関数 heron03 を作成しなさい。ただし，３つの数値から三角形が出来ない場合は999を出力する.

（2）：

```
heron02 <- function(a,b,c){
        if (c < a+b){
         s <- (a+b+c)/2；
         v <- sqrt(s*(s-a)*(s-b)*(s-c))；
       }
         else{
         v <- 999；
       }
return(v)；
}
```

（3）

```
heron03 <- function(a,b,c){
          if (c < a+b && b < c+a && a < b+c ){
              s <- (a+b+c)/2
              v <- sqrt(s*(s-a)*(s-b)*(s-c))
           }
           else{
             v <- 999
           }
         return(v)
}
```

基本問題

〇For 文

問題3　次のような初項から第n項までの和を求める関数sowaを定義しなさい.

（1）sowa1 = 1 + 2 + 3 + 4 + 5 + …

（2）sowa2 = 1 + 2 + 4 + 8 + 16 + …

（3）sowa3 = 1 + 4 + 9 + 16 + …

出力例：sowa1(100)=5050, sowa2(4) = 15, sowa3(3)=14.

基本問題（解答）

問題3（解答）

```
sowa1 <- function(n){
       x <- 0
       for (i in 1:n){
          x <- x+i
       }
       return(x)
    }
```

出力例：sowa1(100)=5050

```
sowa2 <- function(n){
      x<-0
      for (i in 1:n){
        x <- x+2^(i-1)
      }
  return(x)
}
```

出力例：sowa2(4) = 15

```
sowa3 <- function(n){
     x<-0
    for (i in 1:n){
     x <- x+i^2
    }
   return(x)
}
```

出力例：sowa3(3)=14.

応用問題

問題４　次のような初項から第n項までの和を求める関数waを再帰的に定義しなさい.
ただし，if文と再帰的定義を用いること（実習８を参照）

（1）wa1 = 1 + 2 + 3 + 4 + 5 + …

（2）wa2 = 1 + 2 + 4 + 8 + 16 + …

（3）wa3 = 1 + 4 + 9 + 16 + …

出力例：wa1(100) = 5050, wa2(4) = 15, wa3(3) = 14.

応用問題（解答）

関数waの再帰的定義

問題４（解答）

```
wa1 <- function(n){
     if(n==1){
       x <- 1
        }
      else{
       x <- n + wa1(n-1)
        }
     return(x)
}

wa1(100)
```

```
wa2 <- function(n){
     if(n==1){
      x <- 1
      }
       else{
     x <- 2^(n-1) + wa2(n-1)
      }
     return(x)
}

wa2(3)
```

```
wa3 <- function(n){
     if(n==1){
       x <- 1
     }
      else{
      x <- n^2 + wa3(n-1)
     }
     return(x)
  }

  wa3(3)
```

応用問題

問題５　フィボナッチ数列：1，1，2，3，5，8…を生成する関数fibを再帰的に定義しなさい.

出力例：fib(6) = 8

〇カウント関数

問題６　n個の自然数からなるリストに対してその中に偶数がいくつあるかカウントする
関数count_evenを定義しなさい.

出力例：
data <-c (1,2,3,4,5)
count_even(data)=2

応用問題（解答）

問題5（解答）

```
fib <- function(n){
       if(n<=2){
         x <- 1
       }
      else{
         x <- fib(n-1) + fib(n-2)
      }
  return(x)
}



fib(6)
```

問題6（解答）

```
count_even <- function(vec){
      num <- 0
      n <- length(vec)
           for(i in 1:n){
                if(vec[i]%%2==0){
                   num <- num+1
                }
           }
    return(num)
}


data <- c(1,2,3,4,5,6,7,8,9,10)
count_even(data)
```

応用問題

〇線形探索

問題７　すべての値が異なる自然数からなるリストに対し，自然数 n を入力したとき，その数値がリストに入っているかを判定する関数 linear_search を定義しなさい．
入っていればその位置を，入っていない場合は―1を出力する．

出力例

```
data<-c(1,2,3,4,5,6)
linear_search(data,3)=3,
linear_search(data,9)=-1,
```

応用問題（解答）

〇線形探索（解答）

```
linear_search <- function(vec,n){
          len <- length(vec)
          for(i in 1:len){
              if(vec[i]==n){
                  return(i)
              }
          }
      return(-1)
}
```

出力例

```
data <- c(1,2,3,4,5,6)
linear_search(data,3)=3,
linear_search(data,9)=-1,
```

応用問題

〇二分探索

問題８：
値が小さい順に並んでいるリストに対し，このリストから特定の値 n を探索する「二分探索」のプログラム(bin_search)はどのような探索アルゴリズムかを確認しなさい.
（線形探索との比較）

出力例

```
data <- c(2,3,5,7,9,10)
bin_search(data,9)=5
```

応用問題（解答）

二分探索

問題８（解答）

```
bin_search<-function(vec,n){
        left <- 1
        right <- length(vec)
         while(left <= right){
            mid<-floor((right + left)/2)
            if(n == vec[mid]){
                result <- mid
                break
            }
            else if(n > vec[mid]){
                left <- mid+1
            }
            else{
                right <- mid-1
            }
         }
      return(result)
}
```

二分探索は値の小さいグループと値の大きいグループを移動させながら探索範囲を絞り込んで探索していく。

データ群があらかじめ「昇順」や「降順」に整理されていれば、膨大なデータを処理する際は線形探索より効率である.

出力例

```
data<-c(2,3,5,7,9,10)
bin_search(data,9)=5
```

応用問題（解答）

〇ピタゴラス数

問題9（解答）

自然数ｎを入力したとき，ｎまでのピタゴラス数（3つの対）を出力する関数 pythagoras を定義しなさい。（定数倍の重複を許す）

```
pythagoras <- function(n){
      for(i in 1 : n){
          for(j in 1 : i-1){
              for(k in 1 : j){
                  if(k^2 + j^2 == i^2){
                      x  <- c(k,j,i)
                      print(x)
                  }
              }
          }
      }
}
```

出力例：

```
pythagoras(20)
```

応用問題

問題10：Collatz関数

(1) collatz関数（nが奇数→3n+1, nが偶数→n/2）を定義しなさい.

出力例：collatz(9) = 28

(2) collatz関数をs回繰り返すcollatzlist(n,s)を定義しなさい.

出力例：collatzlist(3, 10)=3, 10, 5, 16, 8, 4, 2,1 ,4, 2, 1.

(3) collatzlist関数において1が出力されたら処理を終了する関数collatzlist２を定義しなさい.

出力例：collatzlist２(3)=3, 10, 5, 16, 8, 4, 2,1

応用問題（解答）

問題10（2）（ヒント）　　collatz関数をs回繰り返す関数 collatzlist を作成しなさい。

```
collatzlist <- function(n,s){

   for (          ){                    # 変数をiにして1からsまで繰り返す
       if(n %% 2 ==0 ){
              n <- n/2
       }
        else{
             n <- 3*n+1
        }
        print(n)
   }
}
```

応用問題（解答）

問題10（2）（解答例）

collatz関数をs回繰り返す関数 collatzlist を作成しなさい。

```
collatzlist <- function(n,s){
    for (i in 1:s){
        if(n %% 2 ==0 ){
            n <- n/2
          }
         else{
            n <- 3*n+1
         }
        print(n)
    }
}
```

実行

```
>
collatzlist(4,3)
[1] 2
[1] 1
[1] 4
```

応用問題（解答）

問題10（2）（発展）

出力をベクトル形式にする

```
collatzlist <- function(n,s){
       x <- n                          # ベクトル x に n を代入する
        for(i in 1:s){
           if(n %% 2 ==0 ){
             n <- n/2
           }
            else{
              n  <-  3*n+1
          }
          x <- c(x, n)                 # ベクトル x に順次，結果を追加する
         }
    print(x)
}
```

結果

```
> collatzlist(3,10)
 [1]  3 10  5 16  8  4  2  1  4  2  1
```

応用問題（解答）

問題10（2）（発展）　　外部でcollatzを定義してcollatzlistを定義すると，すっきりする

```
collatz <- function(n){
           if(n %% 2 ==0 ){
             x <- n/2
           }
           else{
             x <- 3*n+1
           }
}
```

```
collatzlist <- function(n,s){
               x <- n
               for(i in 1:s){
                     n <- collatz(n)
                     x <- c(x, n)
               }
          return(x)
  }
```

出力なし関数

応用問題（解答）

問題10（3）

collatzlist関数で1が出力されたら処理を終了

```
collatzlist2 <- function(n){
      x <- n
      while (          ){          # nが１になるまで繰り返す
            n <- collatz(n)
             x <- c(x, n)
        }
     print(X)
}
```

応用問題（解答）

問題10（3）解答例

collatzlist関数で1が出力されたら処理を終了

```
collatz <- function(n){
     if(n %% 2 ==0 ){
          x <- n/2
     }
     else{
         x <- 3*n+1
     }
}
```

```
collatzlist2 <- function(n){
    x <- n
      while ( n > 1){
         n <- collatz(n)
          x <- c(x, n)
      }
      return(x)
}
```

実行

```
> collatzlist3(3)
[1]  3 10  5 16  8  4  2  1
```

発展問題

問題(発展)　　1から100までで一番長い経路をだどって１になる自然数を求めよう

(1)1から100までのcollatz数で1に至るまでのリスト数を定義するcollatzlist3を作成しなさい.

(2) 一番長いcollatz数の経路をグラフ化しなさい.

発展問題（解答）

問題(発展)

1から100までで一番長い経路をだどって１になる自然数を求めよう

```
collatz <- function(n){
     if(n %% 2 ==0 ){
         x <- n/2
     }
     else{
        x <- 3*n+1
     }
}
```

```
collatzlist2 <- function(n){
    x <- n;
     while ( n > 1){
         n <- collatz(n)
          x <- c(x, n)
     }
     return(x)
}
```

```
collatzlist3 <- function(n){
     x <- NULL
        for (i in 1:n){
         y <-length(collatzlist2(i))
         x <- c(x,y)
        }
     return(x)
}

#実行
collatzlist3(20)
```

各自然数の1から20までの経路数

[1] 1 2 8 3 6 9 17 4 20 7 15 10 10 18 18 5 13 21 21 8