Even or odd (easy)

Report whether an Input number is even or odd.

```
Alternate: even/odd
if    remainder  = ▾  0
do    print   Number is even.
else  print   Number is odd.
```

```
Read in number
set input_num ▾ to   prompt for number ▾ with message   Type a number.
```

```
Calculate remainder for odd/even decision
set remainder ▾ to   remainder of   input_num ▾   ÷   2
```
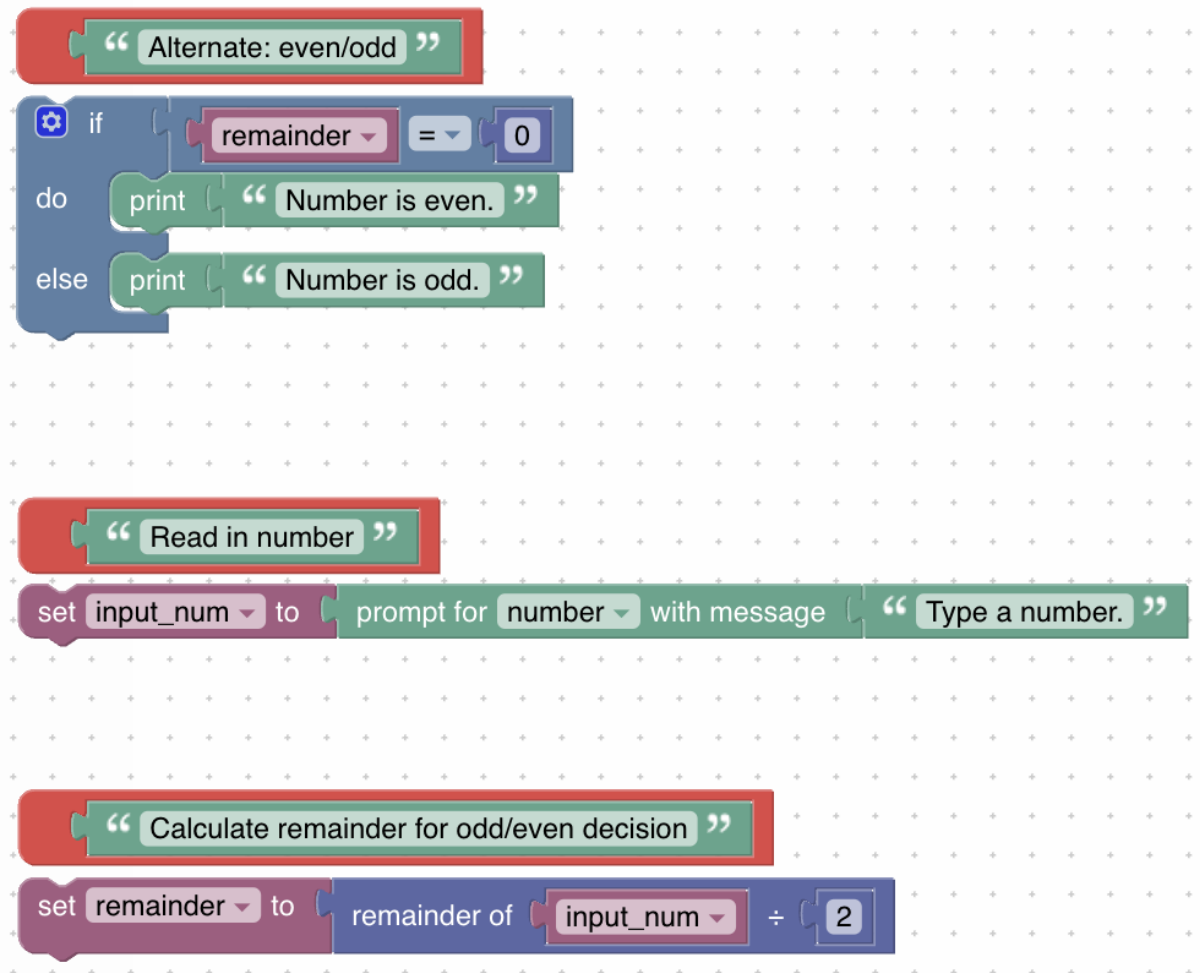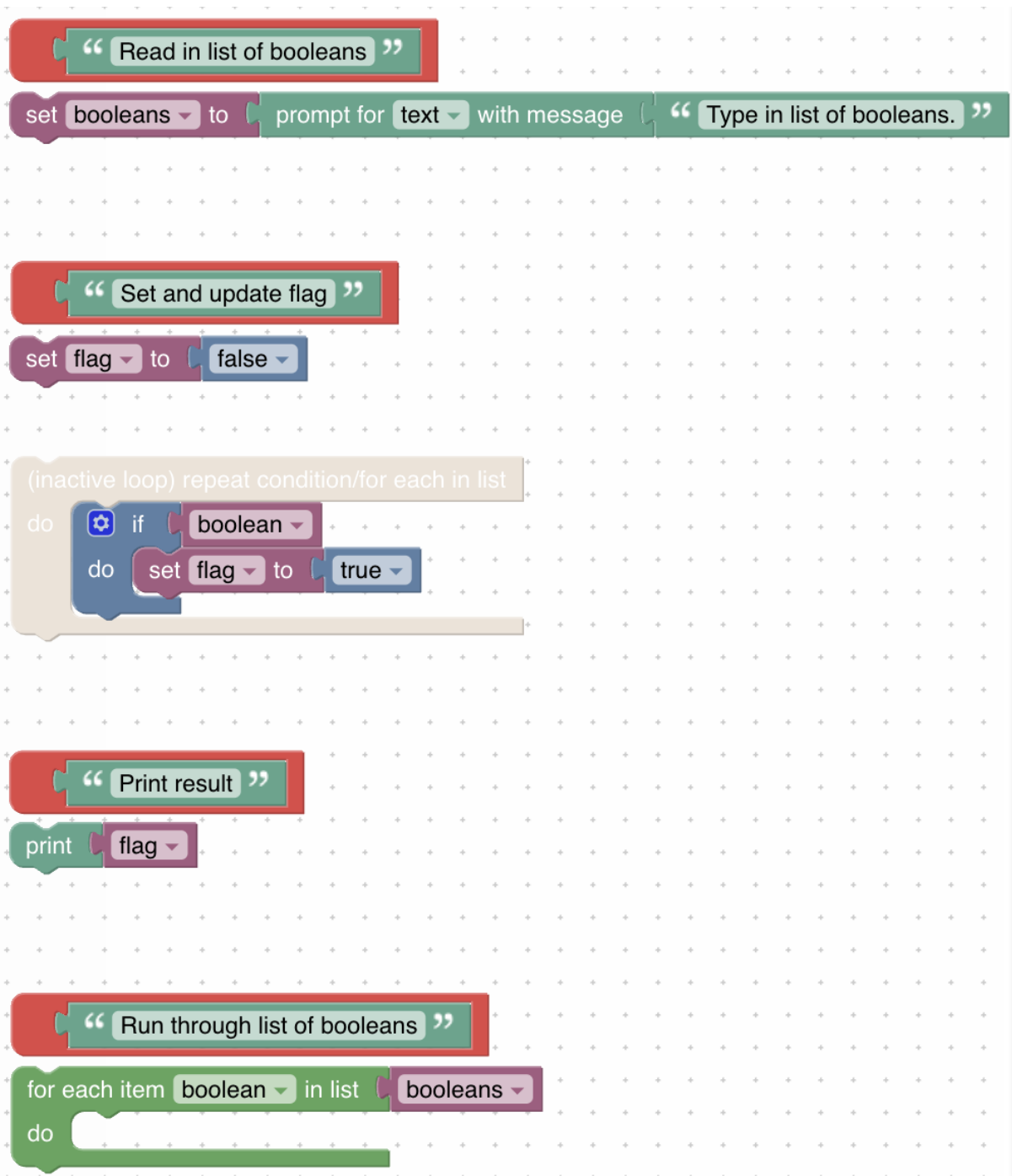
Logical AND (easy)

Write a program that logically evaluates the AND for a list of booleans. If and only if ALL values are True, the program prints True, otherwise False.

```
" Read in list of booleans "
set booleans ▾ to   prompt for text ▾ with message   " Type in list of booleans. "


" Set and update flag "
set flag ▾ to   true ▾


(inactive loop) repeat condition/for each in list
do    ⚙ if    not    boolean ▾
      do    set flag ▾ to    false ▾


" Print result "
print   flag ▾


" Run through list of booleans "
for each item  boolean ▾  in list    booleans ▾
do
```
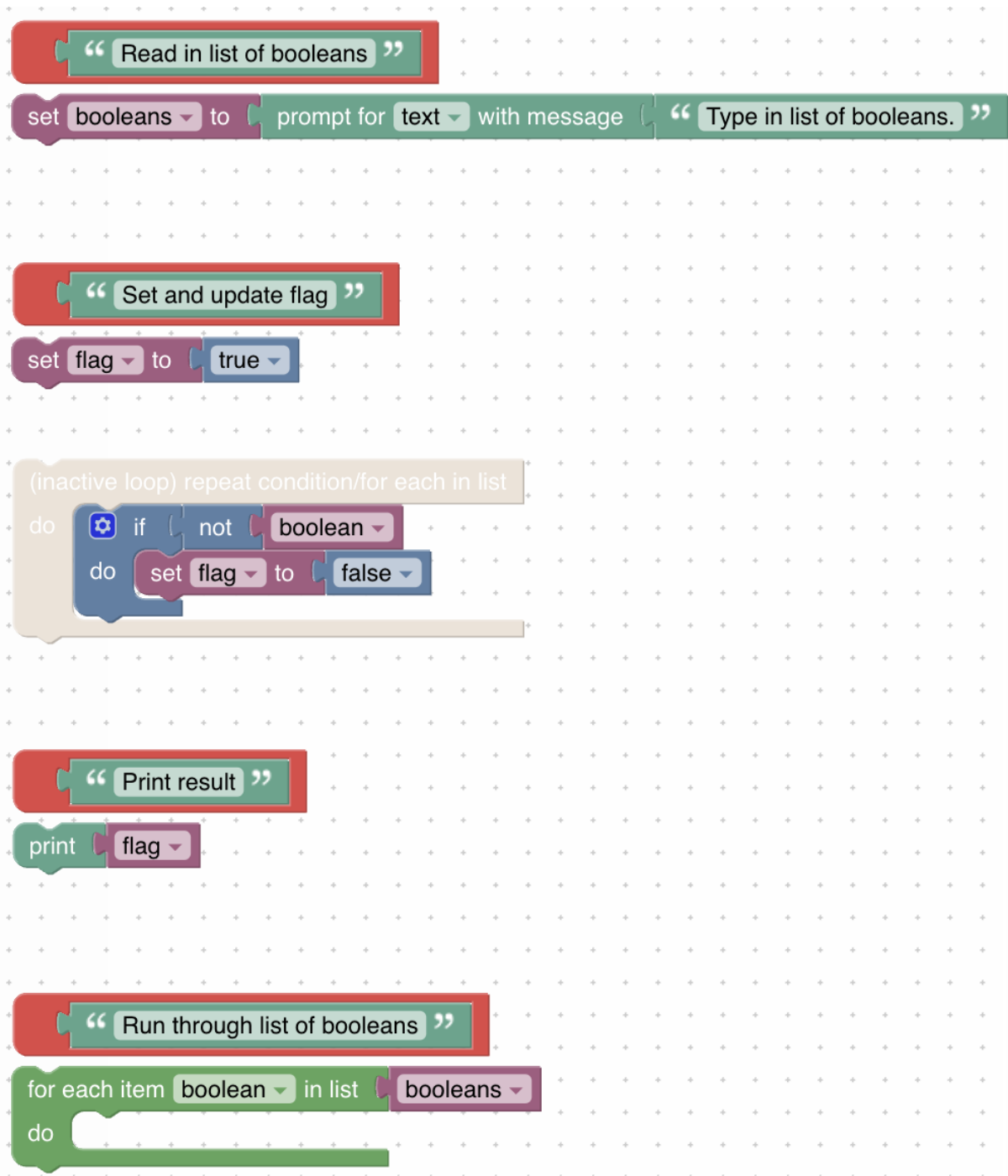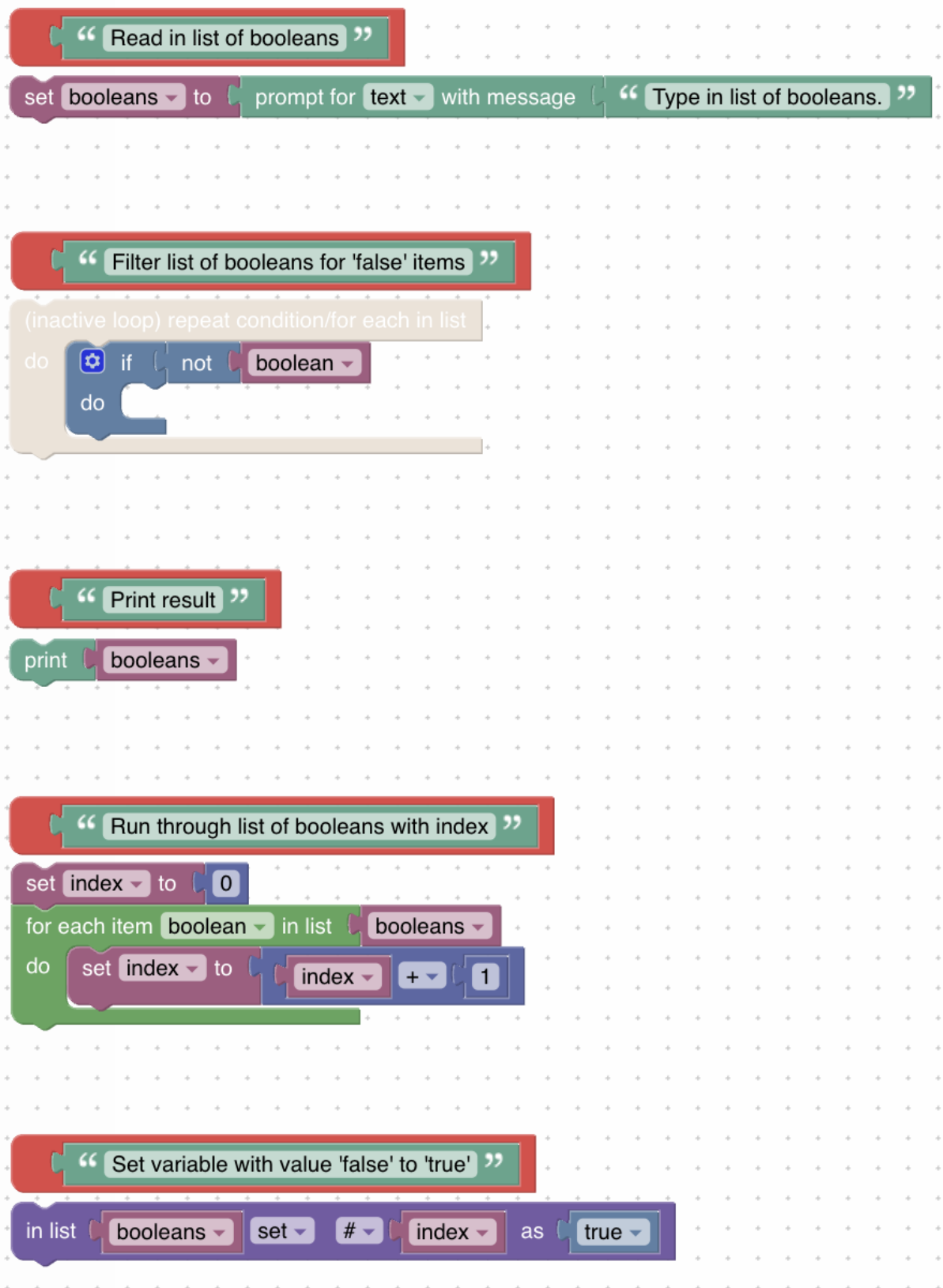
Logical OR (easy)

Write a program that logically evaluates the OR for a list of booleans. If at least one value is True, the program prints True, otherwise False.

```
" Read in list of booleans "
set booleans ▾ to    prompt for text ▾ with message   " Type in list of booleans. "
```

```
" Set and update flag "
set flag ▾ to    false ▾
```

```
(inactive loop) repeat condition/for each in list
do   ⚙ if    boolean ▾
        do   set flag ▾ to    true ▾
```

```
" Print result "
print    flag ▾
```

```
" Run through list of booleans "
for each item boolean ▾ in list    booleans ▾
do
```
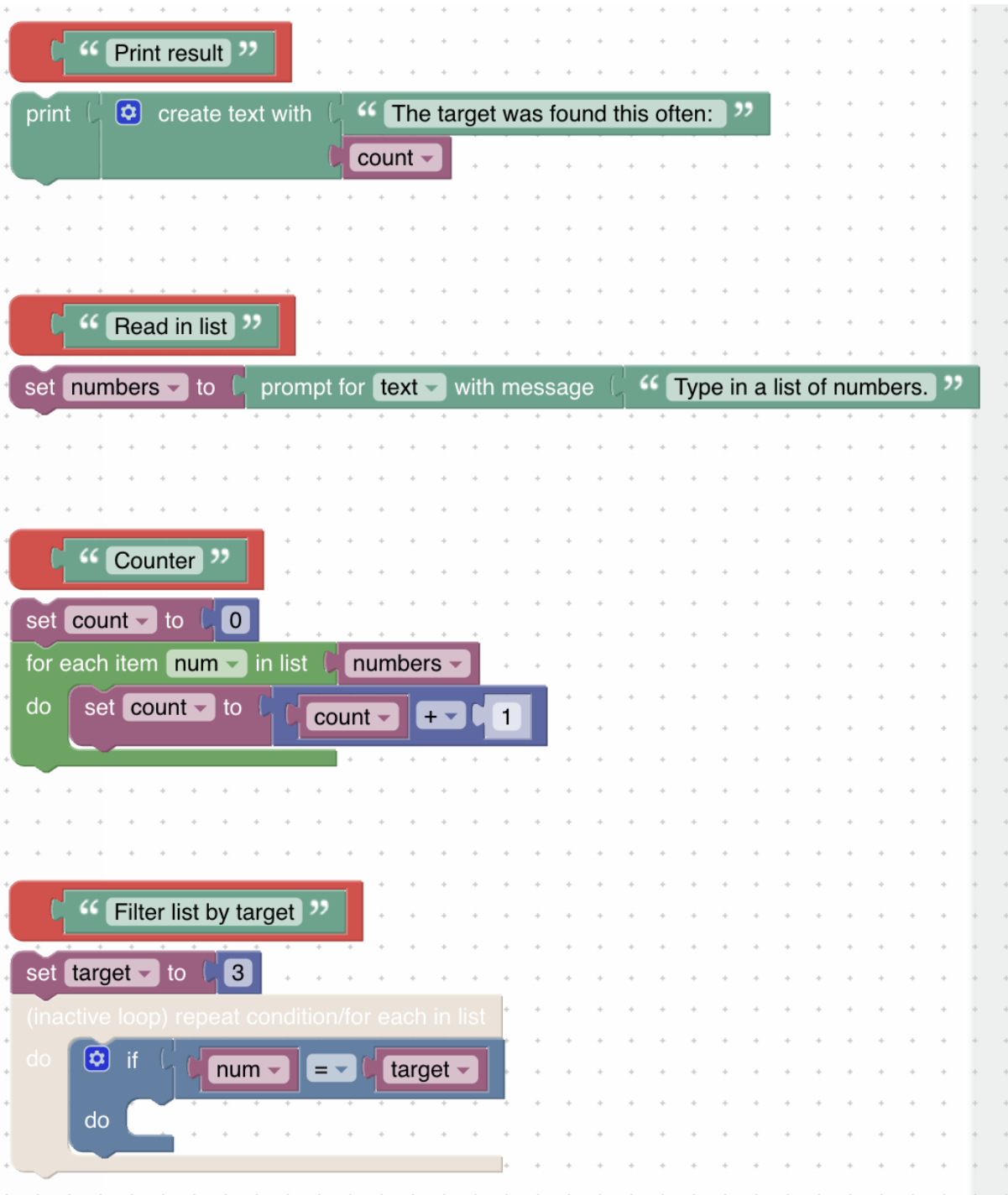
Replace falses (easy)

Write a program that reads in a list of booleans and replaces all occurrences of the value False by True
For example when reading the list [True,True,False] you should in the end print the list
[True,True,True].

Read in list of booleans

set booleans to prompt for text with message " Type in list of booleans. "

Filter list of booleans for 'false' items

(inactive loop) repeat condition/for each in list
do if not boolean
do

Print result

print booleans

Run through list of booleans with index

set index to 0
for each item boolean in list booleans
do set index to index + 1

Set variable with value 'false' to 'true'

in list booleans set # index as true
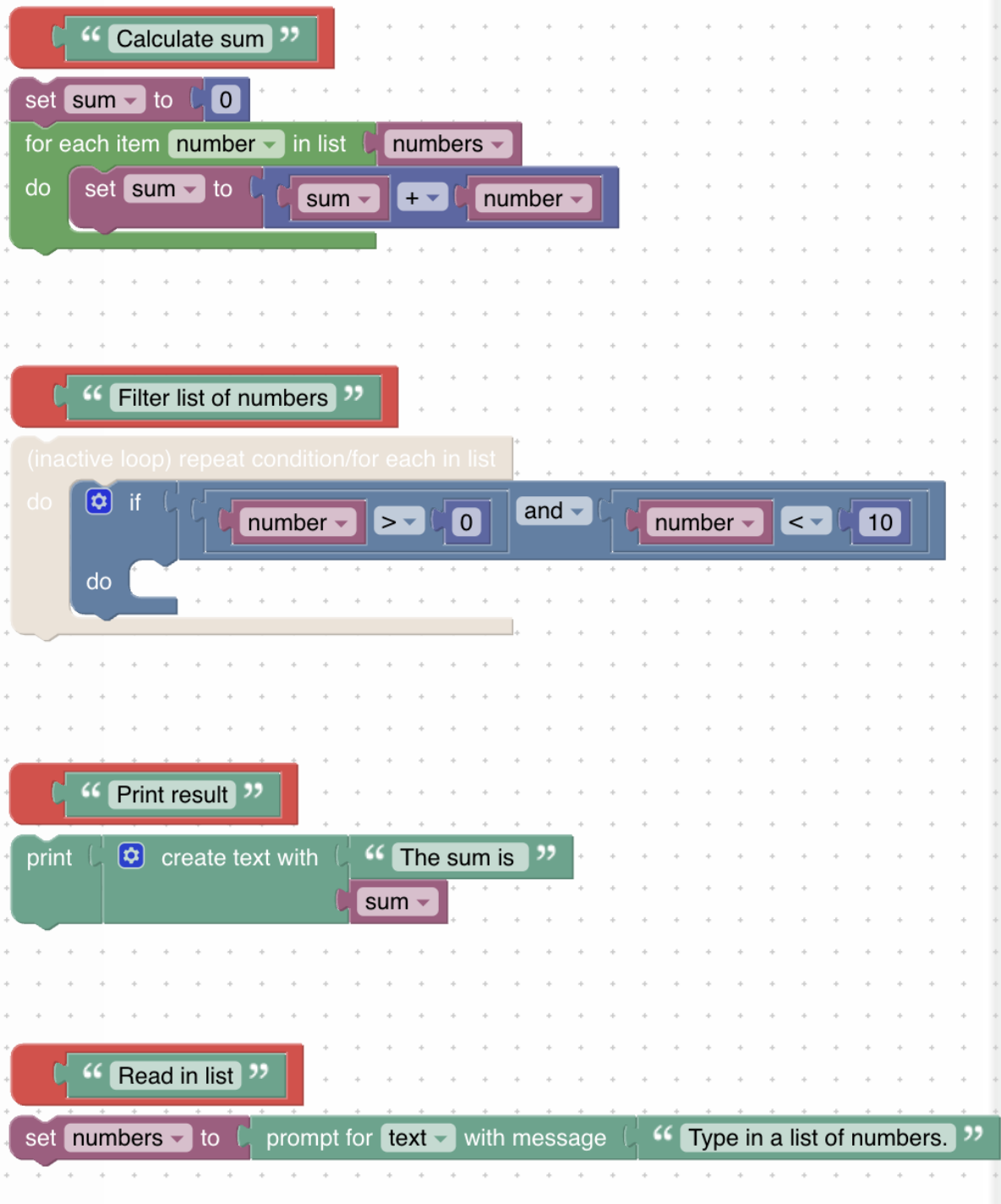
Count target values (easy)

One of the common things to do with a list is to count the number of times a target value appears in a list. Write a program that counts the occurrences of the target value 3 in a list you read in.
For example the program should print out - The target was found this often: 0 - for the list [1, 2, 5] or - The target was found this often: 2 - for the list [3, 4, 3].

```
" Print result "

print   ⚙ create text with   " The target was found this often: "
                              count ▾
```

```
" Read in list "

set numbers ▾ to   prompt for text ▾ with message   " Type in a list of numbers. "
```

```
" Counter "

set count ▾ to   0
for each item num ▾ in list   numbers ▾
do   set count ▾ to   count ▾   + ▾   1
```

```
" Filter list by target "

set target ▾ to   3
(inactive loop) repeat condition/for each in list
do   ⚙ if   num ▾   = ▾   target ▾
     do
```
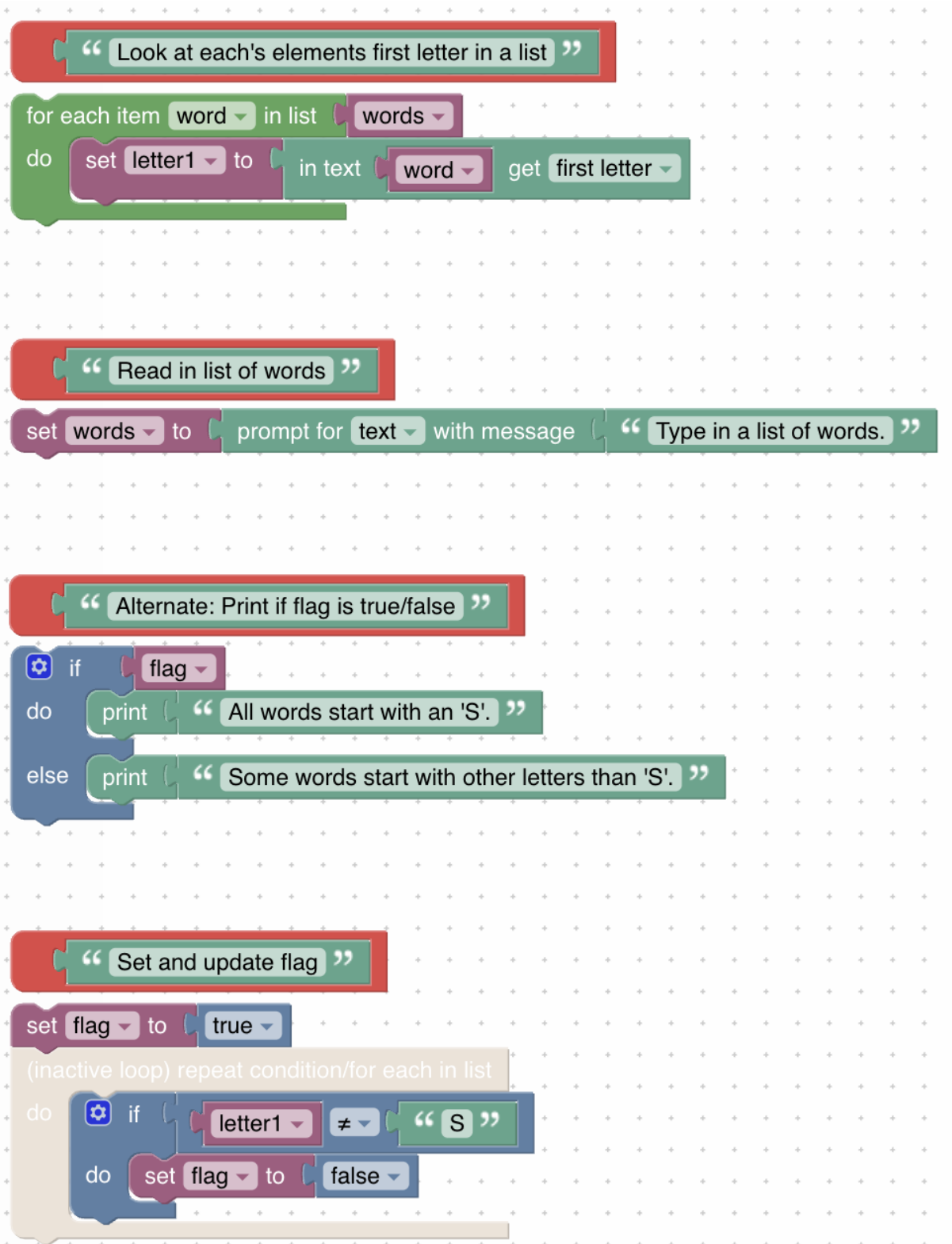
Filtered sum (easy)

Construct a program that reads in a list of numbers and prints the sum of all values in the list that are single digit positive numbers. Loop through all values in the list and add each one that matches the condition to the sum. Print the sum.

```
Calculate sum
set sum to 0
for each item number in list numbers
do   set sum to   sum + number
```

```
Filter list of numbers
(inactive loop) repeat condition/for each in list
do   if   number > 0   and   number < 10
     do
```

```
Print result
print   create text with   The sum is
                           sum
```

```
Read in list
set numbers to   prompt for text with message   Type in a list of numbers.
```
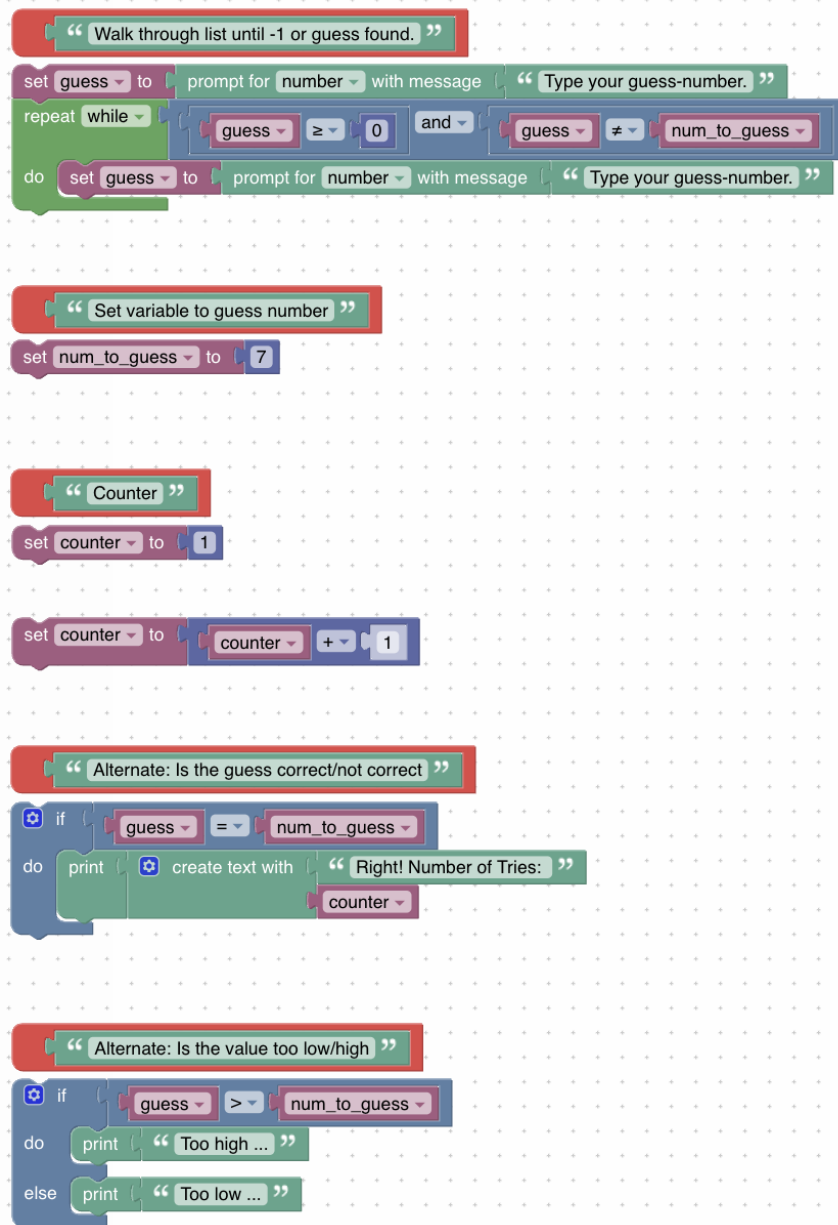
Word start checker (easy)

Construct a program that reads in a list of words and prints out if all words start with the letter S or not. You do not have to worry about upper or lower case – only look for uppercase S.

```
" Look at each's elements first letter in a list "
for each item  word  in list  words
do   set  letter1  to   in text  word   get  first letter
```

```
" Read in list of words "
set  words  to   prompt for  text  with message  " Type in a list of words. "
```

```
" Alternate: Print if flag is true/false "
if  flag
do    print  " All words start with an 'S'. "
else  print  " Some words start with other letters than 'S'. "
```

```
" Set and update flag "
set  flag  to  true
(inactive loop) repeat condition/for each in list
do   if   letter1  ≠  " S "
     do   set  flag  to  false
```

Guessing game (medium)

First we set a guess number (in this case it is set to 7). Now write a program that reads in numbers and ends immediatly when a negative number is read or the guess was correct. Then check if each value matches the guess number. The program should also give feedback if the Input is too low or too high and in the end how many tries it took.
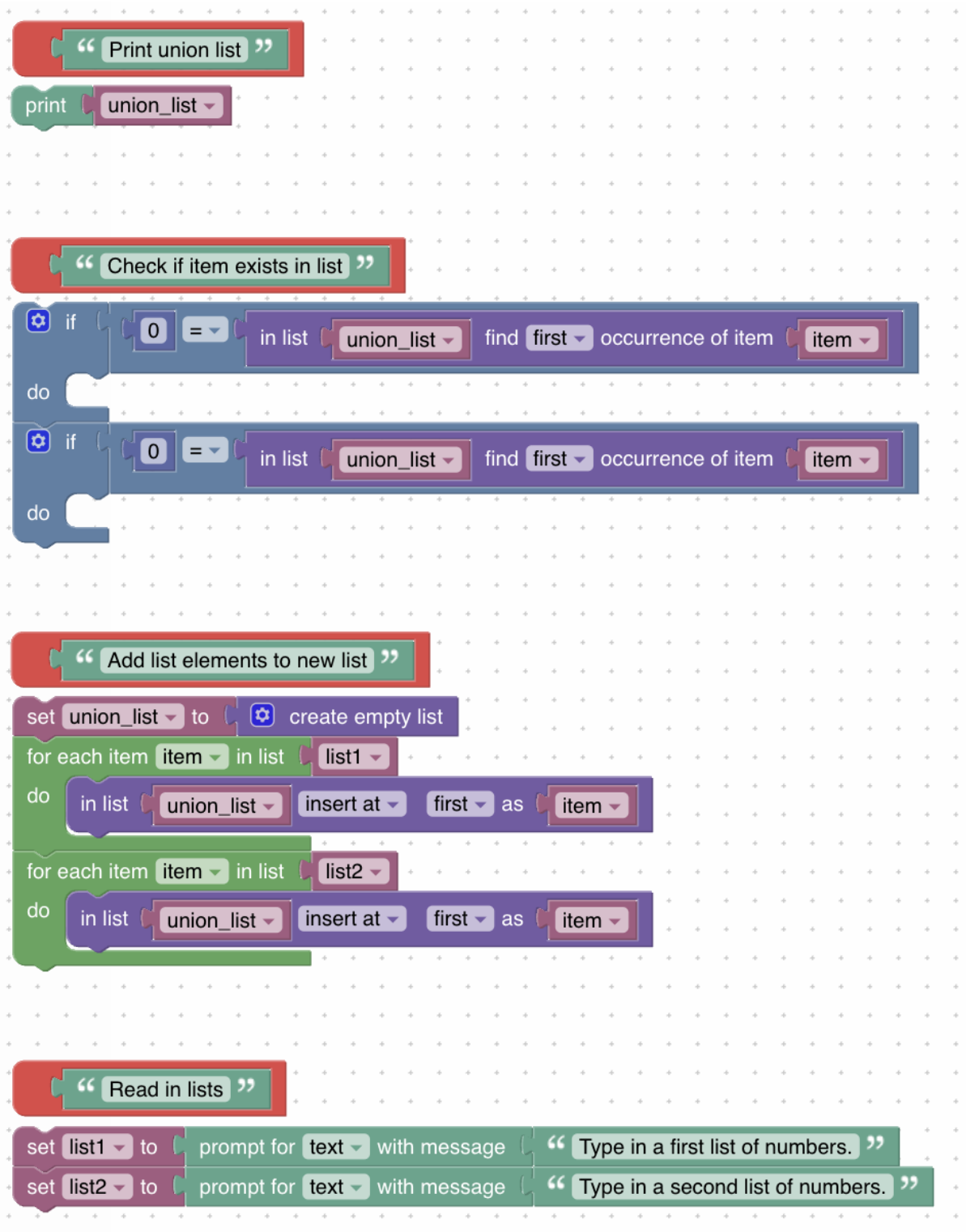
```
" Walk through list until -1 or guess found. "

set guess ▾ to    prompt for number ▾ with message  " Type your guess-number. "
repeat while ▾        guess ▾ ≥ ▾ 0   and ▾   guess ▾ ≠ ▾ num_to_guess ▾
do   set guess ▾ to    prompt for number ▾ with message  " Type your guess-number. "
```

```
" Set variable to guess number "

set num_to_guess ▾ to   7
```

```
" Counter "

set counter ▾ to   1
```

```
set counter ▾ to     counter ▾ + ▾ 1
```

```
" Alternate: Is the guess correct/not correct "

if     guess ▾ = ▾ num_to_guess ▾
do   print     create text with  " Right! Number of Tries: "
                                    counter ▾
```

```
" Alternate: Is the value too low/high "

if     guess ▾ > ▾ num_to_guess ▾
do   print  " Too high ... "
else print  " Too low ... "
```

Union of lists (medium)

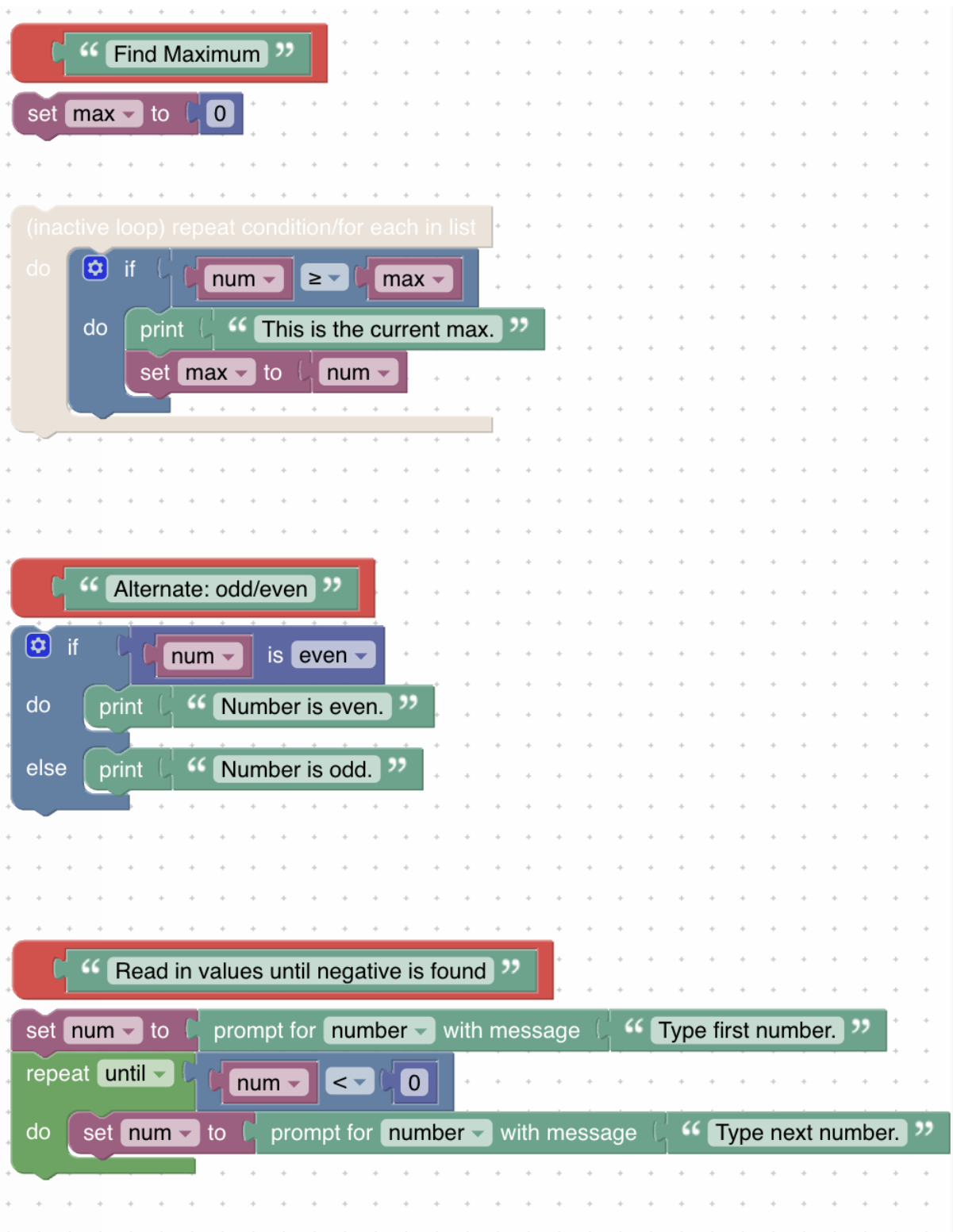Write a program that reads in in two lists and prints the union of both lists.
For example when given the two lists [5,3,7,-1] and [3,7,-5,0] the program prints '[5,3,7,-1,-5,0]'.
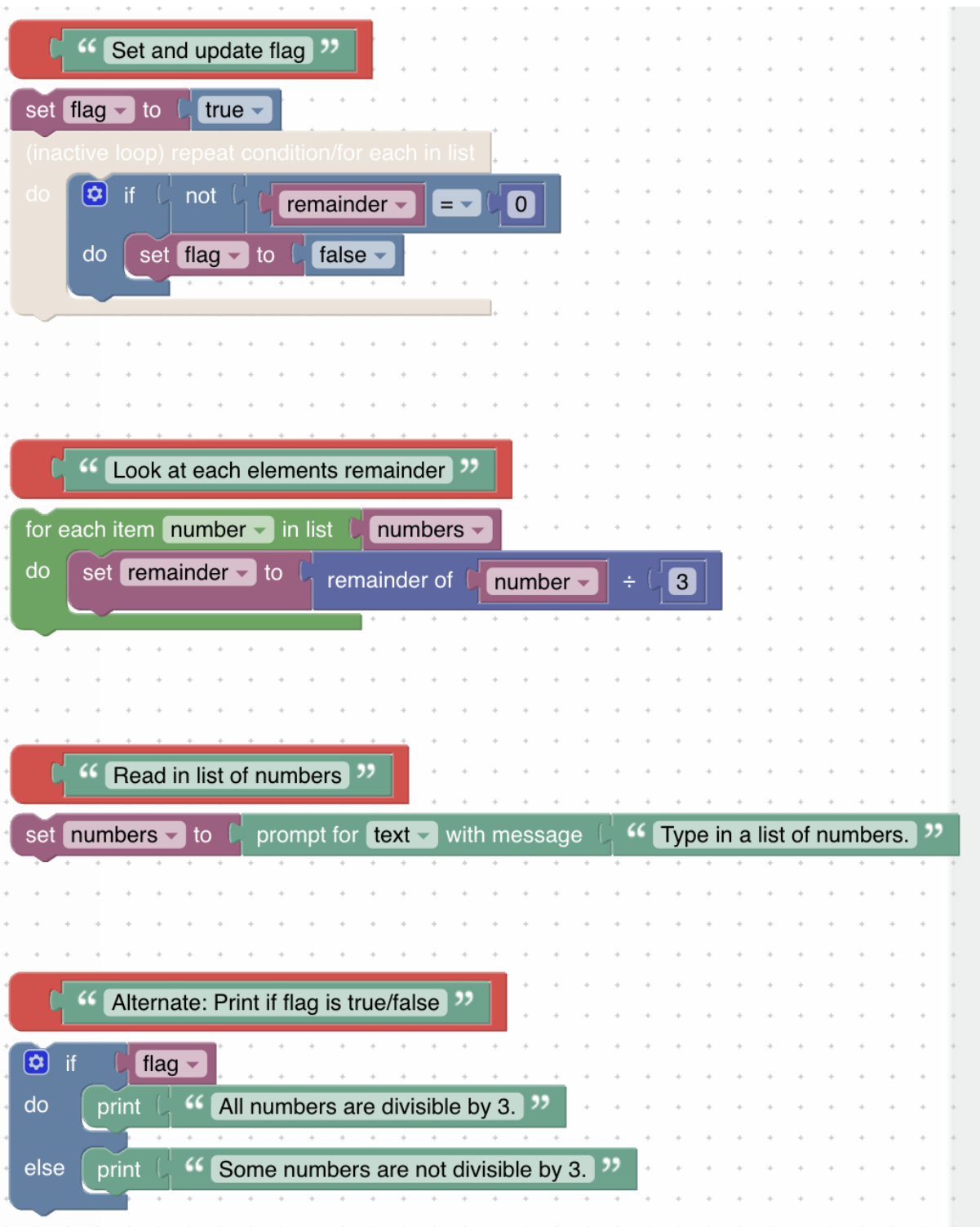Note: The function 'in list [union_list] find first occurance of item [item]' returns 0 if the item is not found.

```
┌─────────────────────────────┐
│  " Print union list "        │
└─────────────────────────────┘
  print  ( union_list ▾ )
```

```
┌──────────────────────────────────┐
│  " Check if item exists in list " │
└──────────────────────────────────┘
  ⚙ if   [ 0 ] = ▾   in list ( union_list ▾ )  find first ▾ occurrence of item ( item ▾ )
  do  [    ]
  ⚙ if   [ 0 ] = ▾   in list ( union_list ▾ )  find first ▾ occurrence of item ( item ▾ )
  do  [    ]
```

```
┌──────────────────────────────────┐
│  " Add list elements to new list "│
└──────────────────────────────────┘
  set ( union_list ▾ ) to   ⚙ create empty list
  for each item ( item ▾ ) in list ( list1 ▾ )
  do   in list ( union_list ▾ )  insert at ▾  first ▾  as ( item ▾ )
  for each item ( item ▾ ) in list ( list2 ▾ )
  do   in list ( union_list ▾ )  insert at ▾  first ▾  as ( item ▾ )
```

```
┌──────────────────────┐
│  " Read in lists "    │
└──────────────────────┘
  set ( list1 ▾ ) to   prompt for  text ▾  with message  " Type in a first list of numbers. "
  set ( list2 ▾ ) to   prompt for  text ▾  with message  " Type in a second list of numbers. "
```

Find even or odd and maximum in list (medium)

Write a program that reads in numbers and ends immediatly when a negtive number is read. For all the numbers in the list it should output first if the number is even or odd and afterwards whether it is the current maximum.

Find Maximum

```
set max to 0

(inactive loop) repeat condition/for each in list
do   if    num  ≥  max
     do  print "This is the current max."
         set max to num
```

Alternate: odd/even

```
if    num is even
do   print "Number is even."
else print "Number is odd."
```

Read in values until negative is found

```
set num to prompt for number with message "Type first number."
repeat until  num < 0
do  set num to prompt for number with message "Type next number."
```

Word start checker B/ Divisible by 3 (easy)

Construct a program that reads in a list of numbers and prints out if all numbers are divisible by 3 (remainder 0) or not.

```
" Set and update flag "
set flag ▼ to   true ▼
(inactive loop) repeat condition/for each in list
do   ⚙ if    not    remainder ▼ = ▼  0
     do   set flag ▼ to   false ▼
```

```
" Look at each elements remainder "
for each item number ▼ in list   numbers ▼
do   set remainder ▼ to   remainder of   number ▼ ÷  3
```

```
" Read in list of numbers "
set numbers ▼ to   prompt for text ▼ with message   " Type in a list of numbers. "
```

```
" Alternate: Print if flag is true/false "
⚙ if   flag ▼
do   print   " All numbers are divisible by 3. "
else print   " Some numbers are not divisible by 3. "
```

Guessing game B/ Guessing Country (easy)

This program should read in suggestions for what you think is my favorite country. First we set my favorite country (Psst ... it is New Zealand). Now write a program that reads in words and ends immediatly when a '-' is read or the guess was correct. Then check if the guess value matches my favorite country. You do not have to worry about upper or lower case, every word will start with an

upper case letter and have otherwise lowercase ones. The program should also give feedback if the guessed country has it's first letter later or earlier in alphabet compared to the correct country and in the end how many tries it took.

```
Set variable to favorite country
set country to " New Zealand "
```

```
Alternate: Is the guess correct/not correct
if       guess_country = country
do  print  create text with " Right! I love New Zealand. Number of Tries: "
                             counter
```

```
Counter
set counter to 1
set counter to   counter + 1
```

```
Alternate: is first letter higher/lower in alpha...
set correct_first_letter to   in text country get letter # 1
set guess_letter to   in text guess_country get letter # 1
if       correct_first_letter > guess_letter
do  print " The correct country starts with a letter later i... "
else print " The correct country starts with a letter earlier... "
```

```
Walk through inputs until '-' or country is found.
set guess_country to   prompt for text with message " Type in what you think is my favorite country. "
repeat while     guess_country ≠ " - "  and   guess_country ≠ country
do  set guess_country to   prompt for text with message " Type in what you think is my favorite country. "
```

Union of lists B/ Intersection of lists (medium)

Write a program that reads in two lists and prints the intersection of both lists.
For example when given the two lists [5,3,7,-1] and [3,7,-5,0] the program prints '[7, 3]'.
Note: The function 'in list [list2] find first occurance of item [item]' returns 0 if the item is not found.

```
" Add list1 elements to intersection list "
set intersection_list to      create empty list
for each item item in list  list1
do    in list  intersection_list  insert at  first  as  item


" Read in lists "
set list1 to   prompt for text with message  " Type in a first list of numbers. "
set list2 to   prompt for text with message  " Type in a second list of numbers. "


" Check if item exists in list2 "
if   [ 0 ≠ ]  in list  list2  find  first  occurrence of item  item
do


" Print intersection list "
print  intersection_list
```

Find even or odd and maximum in list B / Find min and divisible by 3 (medium)

Write a program that reads in numbers and ends immediatly when a negtive number is read. For all the numbers in the list it should output 'This is the current minimum.' when it is the current minimum, and when it is, print also if it is divisible by 3.
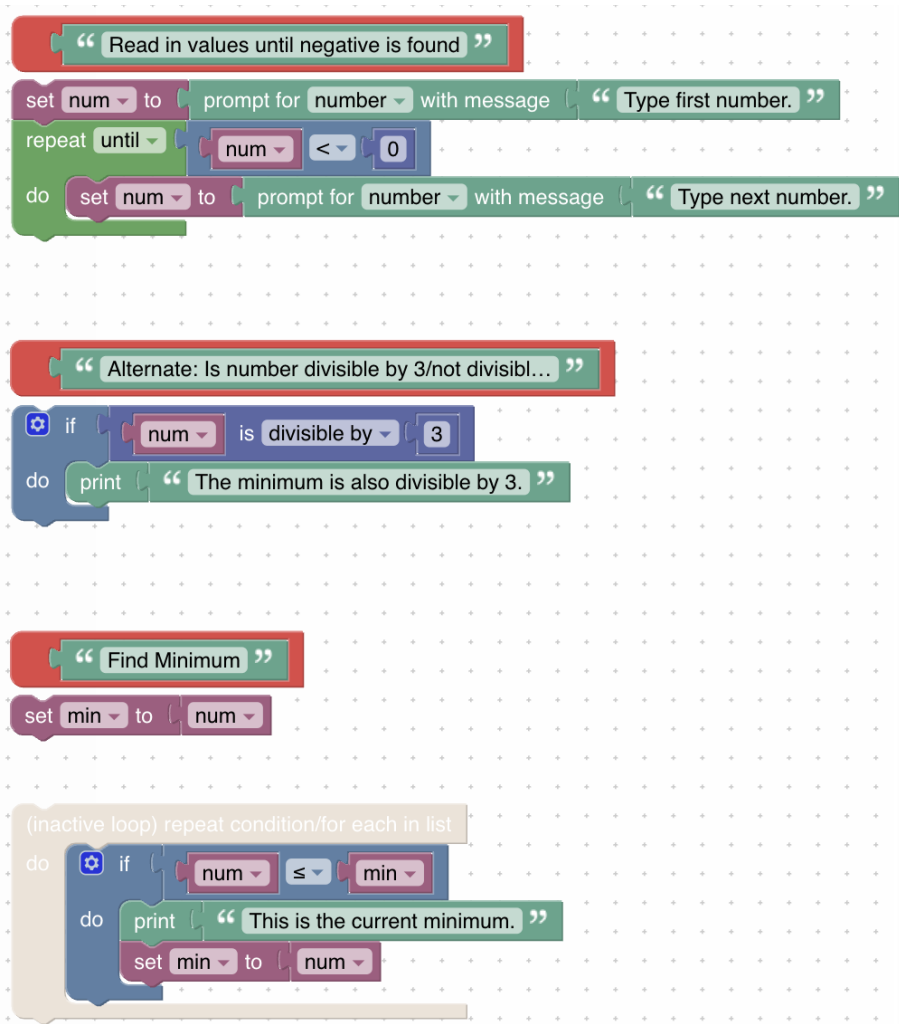
Reverse word and wordlist (medium)

Given a list of words which make up a sentence, generate and print out a string which consists of all the words in reverse order, with each word itself reversed too. Place a space after every word. For example, if the list of words contains "Hello" and "World" in that order, the string created would be "dlroW olleH ".

```
Set variable to list of words

set words to     create list with     Once
                                       upon
                                       a
                                       time
                         Create a list with any number of items.
                                       in
                                       oppositeland
```

```
reverse the letters in a word

set new_word to   " "
for each item letter in list   word
do   to letter append text   new_word
     set new_word to   letter
```

```
Create new list with elements in reverse

set new_words to   " "
for each item word in list   words
do   to new_words append text   new_word
     to new_words append text   " "
print   new_words
```

Average value in range. (medium)

Construct a program that reads a list of numbers. This list consists of a variable number of values followed by an additional number for the start index and a second additional number for the end index of a range of values within the list. The program should print the average of the values between the start and end indices (inclusive). Check if the difference between the end index and the start index is 1 or higher. If so, print the average of the values from start to end and otherwise print 'No avg.' (to prevent a divide by 0).

For example the list [5,3,7,1,7,15,30,2,5] with start 2 and end 5 should print 7.5. NOTE: the indices start from 0 and the end value is included (also not like Python).
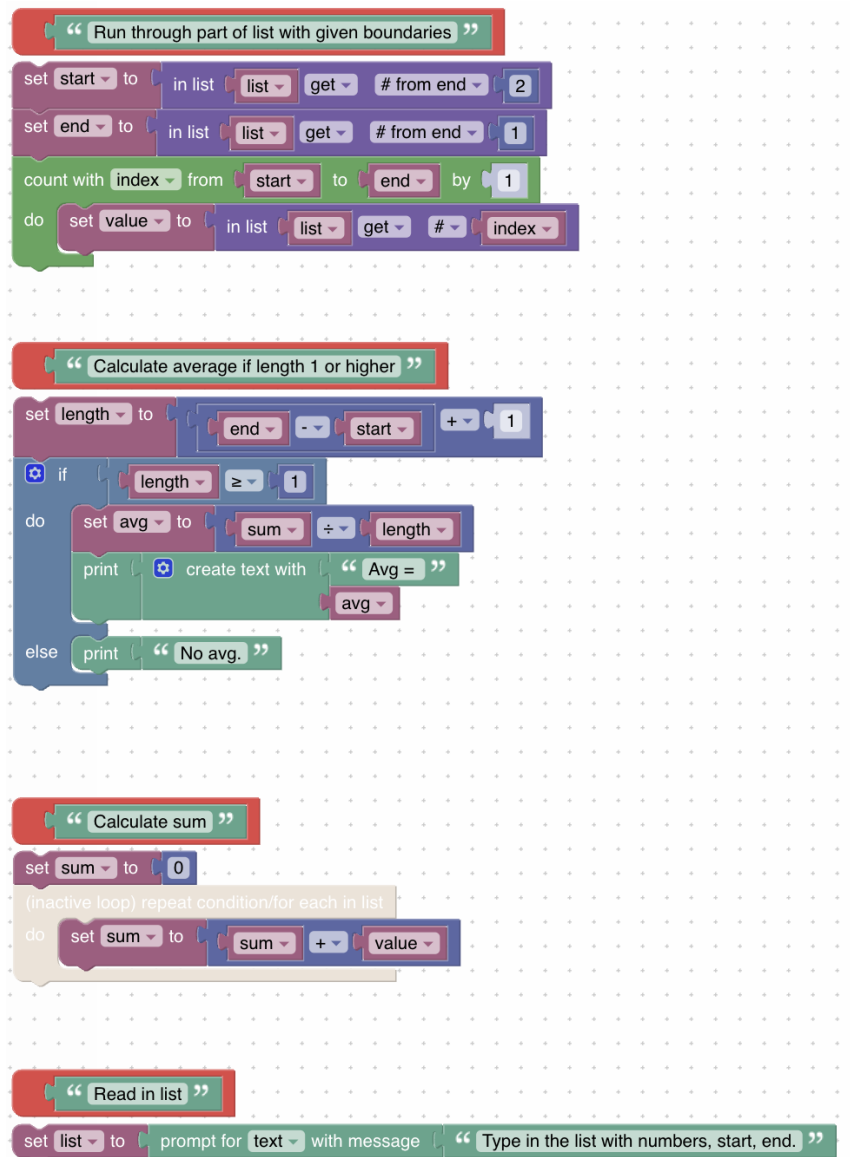
Rainfall problem (advanced)

Let's imagine that you have a list that contains amounts of rainfall for each day, collected by a meteorologist. Her rain gathering equipment occasionally makes a mistake and reports a negative
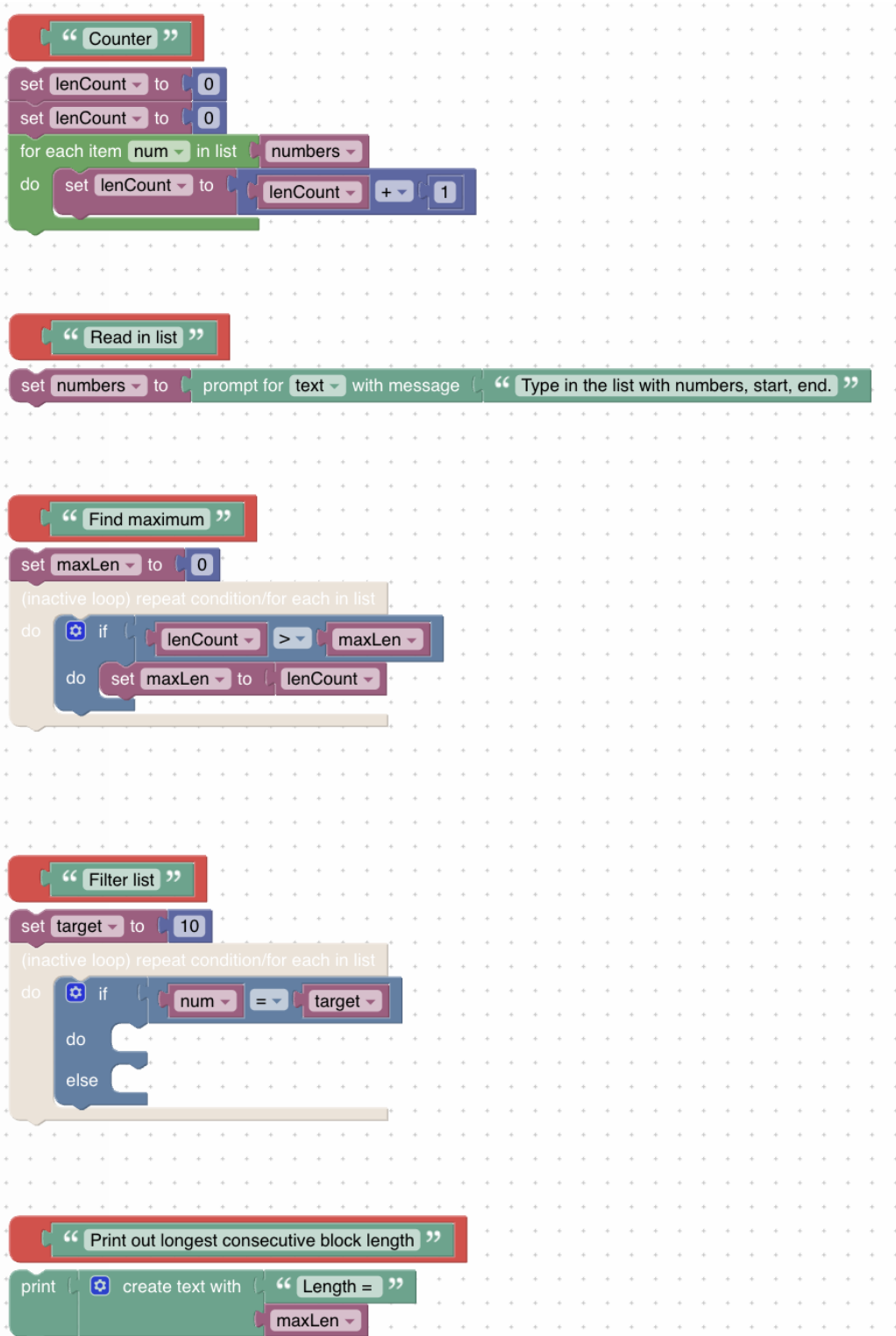
amount for that day. We have to ignore those. We need to write a program to (a) calculate the total rainfall by adding up all the positive integers (and only the positive integers), (b) count the number of positive integers (we will count with "1.0" so that our average can have a decimal point), and (c) print out the average rainfall at the end. Only print the average if there was some rainfall, otherwise print "No rain".
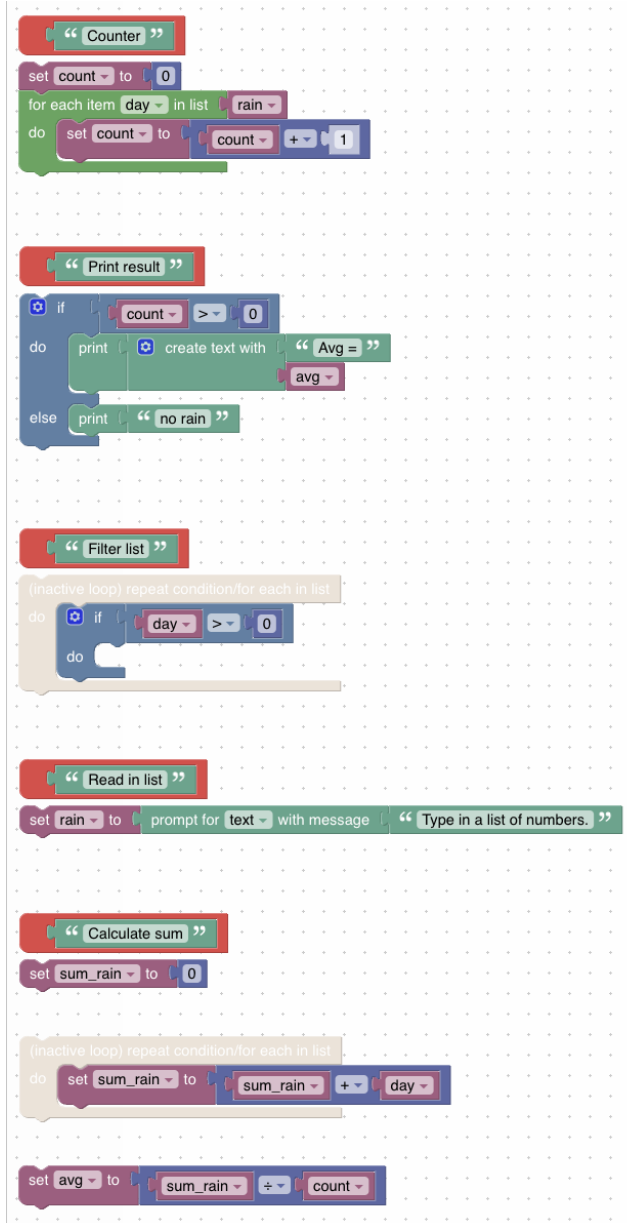
Find longest (advanced)

Write the find-longest program, which aims to find the longest consecutive block of the value target (in this case 10) occurring in the read in list numbers.
For example, if the list numbers contains the values [7, 10, 10, 15, 15, 15, 15, 10, 10, 10, 15, 10] the program should print 3, the length of the longest consecutive block of 10s.

Counter
```
set lenCount to 0
set lenCount to 0
for each item num in list numbers
do  set lenCount to lenCount + 1
```

Read in list
```
set numbers to prompt for text with message "Type in the list with numbers, start, end."
```

Find maximum
```
set maxLen to 0
(inactive loop) repeat condition/for each in list
do  if lenCount > maxLen
    do  set maxLen to lenCount
```

Filter list
```
set target to 10
(inactive loop) repeat condition/for each in list
do  if num = target
    do
    else
```

Print out longest consecutive block length
```
print create text with "Length = "
                       maxLen
```

Is level trail segment (advanced)

A trail segment (for running, walking, cycling, etc) is a list of numbers (known as markers) representing a sequence of heights along the trail. There is always a start and an end marker, and maybe more markers in between.

Write a program to read a list of numbers for a trail segment and print True if the segment is "level"and false otherwise. A trail segment is level if the difference between the maximum and minimum elevation on the segment is less than or equal to 10 metres.

```
"Calculate max"
set max to in list numbers get # start
(inactive loop) repeat condition/for each in list
do    if    value > max
      do  set max to value


"Run through part of list with given boundaries"
set start to 3
set end to 7
count with index from start to end by 1
do  set value to in list numbers get # index


"Calculate min"
set min to in list numbers get # start
(inactive loop) repeat condition/for each in list
do    if    value < min
      do  set min to value


"Read in list"
set numbers to prompt for text with message "Type in a lis of numbers."


"Print result"
print    max + min ≤ 10
```