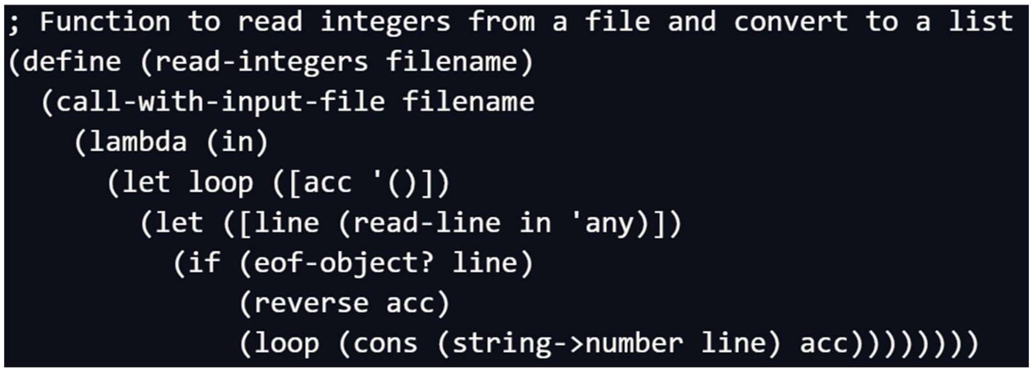
Initial code was generated using the exact copy-pasted requirements for the code in the Program 1 rubric. In this document, final code will have a white background and the previous versions will have the black background. I tried to make minimal manual modifications to the code to see how much the AI keeps itself consistent. It did affect the length of the AI's response because it tended to output the entire program instead of select parts that it had changed.

```
; Function to read integers from a file and convert to a list
(define (read-integers filename)
  (call-with-input-file filename
    (lambda (in)
      (let ([line (read-line in 'any)])
        (map string->number (string-split line))))))
```

This section was revised from a previous code that looked like this:



```
; Function to read integers from a file and convert to a list
(define (read-integers filename)
  (call-with-input-file filename
    (lambda (in)
      (let loop ([acc '()])
        (let ([line (read-line in 'any)])
          (if (eof-object? line)
              (reverse acc)
              (loop (cons (string->number line) acc))))))))
```

This change was due to me not specifying the expected formatting of files being read by this function. The first version had issues reading the files correctly which caused the test code to only return false (#f) instead of a sorted list of integers. After specifying that the files will have integers separated by spaces, CoPilot was able to adjust the function to import the integers correctly. No further action needed to be taken to apply the change to the code.

```
; Function to consolidate the list into value-frequency pairs using procedural hash
(define (consolidate lst)
  (define freq-hash (make-hash))
  (for-each (lambda (num)
              (hash-update! freq-hash num add1 0))
            lst)
  (hash->list freq-hash))
```

This section of code was present in the initial generation and stayed the same until the final version of the code. However, it did disappear from the code temporally when I requested the code to be optimized for processing Data-7, seemingly forgetting the initial requirement I have given. It may have happened because I did not specify which parts it was not allowed to modify.

```
; Function to find the minimum and maximum values in a list
(define (min-max lst)
  (foldl (lambda (x acc)
           (let ([current-min (car acc)]
                 [current-max (cdr acc)])
             (cons (min x current-min) (max x current-max))))
         (cons (car lst) (car lst))
         lst))
```

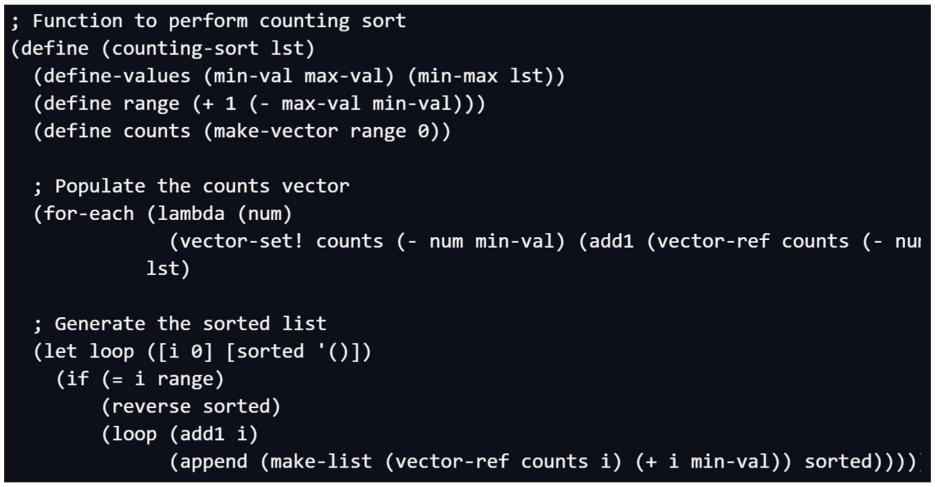
This function was not present in the initial generation and was added later when I specified no sorting library should be used and also the sorting speed for Data-7 should be faster. From this function I can see that AI was not keeping track of the outputs as it was only returning 1 output (either min or max value) without the other, even when the function comment it generated suggest it should be returning 2. However, when I showed it the error message, it was able to locate and fix the issue in one go, without me specifying which function the error message originated from.

```
; Function to perform counting sort without library sort function
(define (counting-sort lst)
  (let* ([min-max-pair (min-max lst)]
         [min-val (car min-max-pair)]
         [max-val (cdr min-max-pair)])
    (define range (+ 1 (- max-val min-val)))
    (define counts (make-vector range 0))

    ; Populate the counts vector
    (for-each (lambda (num)
                (vector-set! counts (- num min-val) (add1 (vector-ref counts (- num min-val)))))
              lst)

    ; Generate the sorted list
    (let loop ([i 0] [sorted '()])
      (if (= i range)
          (reverse sorted)
          (loop (add1 i)
                (append (make-list (vector-ref counts i) (+ i min-val)) sorted))))))
```

As predicted, AI tried to implement the counting sort using the sort library:



```
; Function to perform counting sort
(define (counting-sort lst)
  (define-values (min-val max-val) (min-max lst))
  (define range (+ 1 (- max-val min-val)))
  (define counts (make-vector range 0))

  ; Populate the counts vector
  (for-each (lambda (num)
              (vector-set! counts (- num min-val) (add1 (vector-ref counts (- nu
            lst)

  ; Generate the sorted list
  (let loop ([i 0] [sorted '()])
    (if (= i range)
        (reverse sorted)
        (loop (add1 i)
              (append (make-list (vector-ref counts i) (+ i min-val)) sorted))))
```

(won’t let me see the entire line, but it can be viewed in the chatlog)

I can see that AI was aware of which parts of the code were responsible for the counting sort, and what kinds of input the rest of the function needed as they were left unmodified. The AI dividing the function into different parts also makes it appear that it is aware of their purpose. I've noticed that AI is at least knowledgeable of when and where the task should be divided into smaller functions, as well as organize it with comments.

```
; Function to check if a list is sorted in ascending order
(define (is-sorted? lst)
  (or (null? lst)
      (null? (cdr lst))
      (and (<= (car lst) (cadr lst)) (is-sorted? (cdr lst)))))

; Function to test with multiple files
(define (test-multiple-files filenames)
  (for-each
   (lambda (filename)
     (let* ([integers (read-integers filename)]
            [sorted-list (counting-sort integers)]
            [sorted (is-sorted? sorted-list)])
       (unless (string=? filename "Data-7.txt")
         (displayln (string-append "File: " filename))
         (displayln sorted-list))
       (displayln (string-append "Sorted: " (if sorted "Yes" "No")))
       (newline)))
   filenames))

; Main function to test given text files
(define (main)
  (test-multiple-files '("Data-1.txt" "Data-2.txt" "Data-3.txt" "Data-4.txt" "Data-5.txt" "Data-6.txt" "Data-7.txt"))
  (is-sorted? '(1 3 2 4 5)))

; Call the main function
(main)
```

These are the functions that helped me test the inputs in the data files. It's interesting how the AI put all the input file names into a single list, instead of running the test function for each one individually. It might have been caused by my wording where the AI thought I was grouping the files together as if they are related to each other. Note that exclusion for displaying the Data-7 file contents were intentionally disabled as I didn't want to print millions of integers (my patience ran out after the 5th execution).

Regarding cycling error messages (in my other attempt):

This was my second attempt at generating a working code. I gave up on the first attempt when the AI ping-pong'd back and forth between 2 output errors (total of 7 times, if I remember correctly), and refused to make it work even after giving it a suggestion. Considering how effective it was at solving errors here (if you check the chat log), I'm divided on the AI's capabilities for locating and resolving bugs. It might have simply lacked data for adjusting outputs for that specific function.