EXERCISE 2

Read Text Record

```
read_text_record("records.out.01")

INTEGER = 1
INTEGER = 2
INTEGER = 3
INTEGER = 4
INTEGER = 5
Read_text_record completed.
```

Objectives:

- Write a function to open an external file with a variable name.
- Read the contents of the external file into a virtual array.
- Dump the contents of the array

Read Text Record

Exercise Description:

The function, `read_text_record( filename )`, will open an external file called *`filename`* and read the integer values on each line into an array. The function will check on the success of the file opening operation and exit with error messages in case of problems. The function will need to read each record of *`filename`* for integers until an end of file is encountered. The integers will be placed in an array of numbers. After the file is read completely, the function verifies the contents of the array. Finally, it closes the external file, clears the memory, and tells the user the function has finished executing.

To test this function, we will use the file created in the previous exercise. Therefore, `filename = record.out.01`, the command window should read:

```
INTEGER = 1
INTEGER = 2
INTEGER = 3
INTEGER = 4
INTEGER = 5
```

Files:

All the files that used in this exercise are listed below. Each list includes the file, where it originated, and a summary of information of how it relates to the exercise.

File	Supplied/Created	Description
read_text_record.pcl	Created	A file that should be able to search a file for a string pattern and then count the number of instances in which that string pattern occurs.
records.out.01	Created	Should be created with the number of records that are written in the PCL file.

Exercise Procedure:

1. Enter the vi editor and create a PCL function in a file called `read_text_record.pcl`.

As you are creating these PCL functions, it would be nice to have them all in one directory, like a `bin` directory. But you may not want to run MSC⁄PATRAN from this `pcl` directory all the time. By default, PATRAN will search your current working directory for the PCL files. However, one can add to this search path using the following directive:

```
!!PATH directory
```

`directory` is a relative or absolute path to the PCL directories.

2. Compile the function.

   Start the PCL compiler by typing:

   ```
   %p3pclcomp
   ```

   in your xterm window.

Enter the command:

```
!!input read_text_record.pcl
```

into the MSC⁄PATRAN PCL compiler. The prompt should look like the one shown below.

```
P3/PCL Compiler Version 5.0
Exit or ctrl-d to quit
-> !!input read_text_record.pcl
Compiling: read_text_record
 Compiled: read_text_record
->
```

All the error messages and diagnostics will be written to the xterm.

3. Test the function.

   If the name of your output file is `records.out.01`, then type the command:

```
->read_text_record("records.out.01")
```

in the xterm p3pclcomp prompt.

Read Text Record

You should get the following:

```
INTEGER = 1
INTEGER = 2
INTEGER = 3
INTEGER = 4
INTEGER = 5
Read_text_record completed.
->exit
```

Sample Solution:

```
FUNCTION read_text_record( filename )


/* Purpose: This function opens an external file and reads
 * the integer values on each line into an array.
 *
 * Input: filename S name of external file
 *
 * Output: none
 *
 */

   STRING filename[]

   STRING record[720], field[128]
   INTEGER num_ints, status, channel, itoken, array_alloc_size
   INTEGER lrecl, increment = 1000, i
   INTEGER numbers( VIRTUAL )

   /*
   * Open the external file
   */

   status = text_open( filename, “OR”, 0, 0, channel )
   IF( status != 0 ) THEN
          write(“Unable to open file “//filename)
          RETURN status
   END IF

   /*
   * initialize variables and counters
   */

   array_alloc_size = increment
   sys_allocate_array( numbers, 1, array_alloc_size )
   num_ints = 0

   /*
   * Loop to end of file reading all records
   */


   WHILE( text_read_string( channel, record, lrecl ) == 0 )

   /*
   * Check for integers and blank tokens
   */

          itoken = 0
          REPEAT
                 itoken += 1
                 field = str_token( record, “ “, itoken, TRUE)
                 IF( str_datatype( field ) == “INTEGER”) THEN
                        num_ints += 1
                        IF( num_ints > array_alloc_size ) THEN
                               array_alloc_size += increment
                               sys_reallocate_array( numbers, 1, array_alloc_size )
                        END IF
                        numbers(num_ints) = str_to_integer( field )
                 END IF
          UNTIL ( field == ““ )
```

```
    END WHILE


    /*
    * Verifing the array prior to release of the memory
    */

    FOR (i= 1 to num_ints)
           dump numbers ( i )
    END FOR

    /*
    * Close the external file and release memory
    */

    text_close( channel, “ “ )
    sys_free_array( numbers )

    /*
    * Tell user complete
    */

    ui_write(“Read_text_record completed.”)


END FUNCTION
```