Assignment 05 – Sorting an array of strings in PEP/8$_2$ (Due: Part I - Friday April 22nd, Part II - Monday April 25th 2016)

For this assignment you are to complete the given PEP/8$_2$ source program, `SortStrings.pep2`, that reads an "empty string terminated" sequence of string values into an array and then calls upon a method to sort these element values. That is, it rearranges the structure so that the strings may be readily printed in lexicographic ascending order.

Although this task is very similar to the one for Assignment 04, a major difference has to do with the data structure used to represent the strings. Since the `STRI` macro instruction has been developed to operate with a "string object" it is preferable to keep both an array of string objects (to store the strings) and a parallel array of references (i.e. pointers) into this array for each of the strings. Recall, that a "string object" here is a sequence of bytes allocated to represent the characters in the string immediately preceded by a "before byte" descriptor that stores the capacity (not the length; that's a different measure) of the following sequence.

In the following excerpt from `SortStrings.pep2`, `LENGTH` and `LIMIT` are given for information purposes, the "chevrons" and "dashes" are declared so that when memory is looked at it will be easier to locate the array of pointers and the array of strings. `p` then is the array of pointers and `a` is the array of strings. See screen shot given later.

```
;--------
LENGTH:    .EQUATE   31           ; Expresses the maximum length of a string
LIMIT:     .EQUATE   24           ; Expresses the capacity of the array
           .ASCII    "<<<<<<<<"   ; Just a marker for visual inspection
p:         .BLOCK    48           ; The array of pointers; LIMIT * 2
           .ASCII    "--------"   ; Just a marker for visual inspection
a:         .BLOCK    768          ; The array of strings; (1+LENGTH) * LIMIT
           .ASCII    ">>>>>>>>"   ; Just a marker for visual inspection
count:     .BLOCK    2            ; The number of strings read and stored
;--------
```

Thus the sort subprogram you develop will not be moving the string objects around, but rather will be rearranging the references (in the `p` array). Since I consider this to be a more involved and more challenging assignment than Assignment 04 was, this assignment is presented in two parts.

Part I – Develop and test `ScompTo`

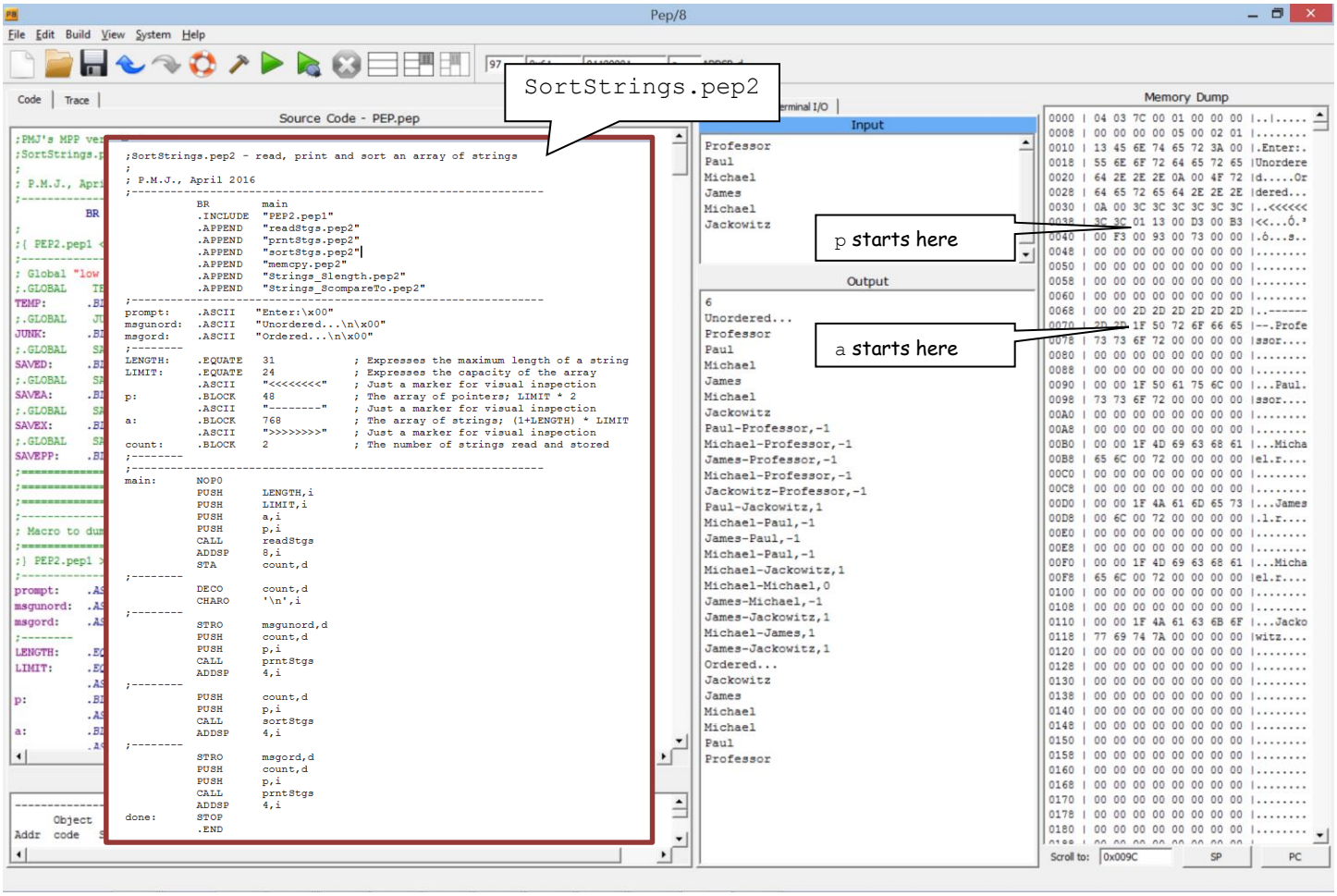
Develop the subprogram, `ScompTo`, that operates as a Java `compareTo` function in that it compares two string objects and returns a negative, zero or positive number indicating the relationship between the two string values being compared. Also develop a tester program, `TestScompTo.pep2`, (a throwaway) with a loop that reads two strings, calls upon `ScompTo` to compare them, prints out the result and continues this until a pair of empty strings is entered. The `Slength` subprogram should serve as a useful example for you to follow in developing `ScompTo`.

Part II – Develop and test `prntStgs` and `sortStgs`

Develop the currently stubbed `prntStgs` (printStrings) and `sortStgs` (sortStrings) subprograms used in `SortStrings.pep2`, and test this program until you are satisfied that it fully performs the indicated task. Note that since you will by then have a reliably tested means of comparing two strings this should fall together nicely.

Consider closely the following screen shot showing my working solution for this assignment using a small set of data. In this case seven strings are read (including the empty string) and these are then printed in unordered and ordered fashion. This version of the program has been "instrumented" in that it contains instructions to print out

some informative information depicting the results of string comparisons that have been make. As with my Assignment 04 solution, the Selection Sort Algorithm has been employed. I draw your attention to the annotations that point out some observations visible in the memory dump as well.



You are to use the "submission form" provided for Assignment 05 on the Course Web Site (CWS) to submit your work on this assignment. For Part I please submit the files `ScompTo.pep2` and `TestScompTo.pep2` along with a MS Word document file (to be named `PartI.docx`) that presents screen shots (and accompanying explanations and annotations) that illustrate the testing you have done on this part and (hopefully) also illustrating that the subprogram works correctly. For Part II please additionally submit the files; `prntStgs.pep2`, `sortStgs.pep2` and your corresponding revised and completed `SortStrings.pep2`. You are encouraged to likewise prepare and submit another MS Word document file (to be named `PartII.docx`) in which you present screen shots and explanations of this part of the assignment.

Of course, all source program files must be appropriately and usefully commented. In particular, there must be a comment block at the beginning of each file that (a) indicates that it is part of a solution to Assignment 5 in CMPS 250 for Spring 2016, (b) identifies the person who developed the program and is submitting it, (c) acknowledges all persons (by their full names) who collaborated with the submitter in completing this assignment (or state that you worked alone), and (d) point out any flaws or deficiencies in the program of which you are aware.

Good luck, P.M.J.