HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF TELECOMMUNICATIONS ENGINEERING

COMPUTING SYSTERMS AND PROGRAMMING
LC- 3 PROJECT

Lecturer: **Assoc. Prof. Đặng Thành Tín**

Students : **Trần Minh Trí _ 2051022**

TOPICS

PROJECT: LC-3 programming

This takes you 10% of overall score on my side

The requirement for the project:

1. Use subroutines as much as possible
2. Create user interface as clear and beautiful as possible
3. Check range for every value input and output appropriately
4. The program should be organized so well for structure programming
5. The program needs to comment as mamy as possible
6. Each student must choose only one topic
7. The file .asm of traditional name (StuID_name…) must be submitted to BKel right before final exam including all supplementing files if you want to make clear more about the project, such as results output, …

Write a program to input *n* (input from keyboard) strings of characters with the length unlimited (it is defined by the program, not by the compiler). Sort them in descending or ascending order depending on the request input.

Attention:

These strings are sorted in ascending/descending by the order of dictionary with deleting redundant characters in each string: blank ‘ ‘, comma ‘,’ if they exist in string.

<u>I/ MAIN ALGORITHM</u>

1. Enter number of n strings and enter characters

+ Create a command that requires entering the number of strings for the user with the STRINGZ, PUTS command (the program reports ERROR when the user enters the ENTER key when the number has not been entered).

+ Create commands to input and output characters for each string with GETC, OUT . commands

+ Save the characters just entered into the memory cell x5000 or later for next use

2. Remove extra space and comma characters internally in each string

2.1 Remove extra spaces and commas if present at the beginning of each string (using subroutine)

2.2 Remove extra spaces and commas inside the string (using subroutine)

2.3 Remove the extra characters at the end of each string and append a "0" to the end of each string to use the output command in the last step.

3. Count the total number of string characters and store the address of the first characters of each string

+ Count the total number of characters of the string

+ Use the total number of characters of the string to find the address of the first character of each string

+ Save the first character of each string to x6000

4. Sort the order between strings

+ Use Bubble Sort algorithm to sort

 Note: compare each character in each string and then sort the addresses of the first characters (using Bubble) in ascending order based on comparing characters between strings

5. Output the strings in order

+ Get each address value of the first characters in each sorted string

+ Use the PUTS command to output characters based on the first characters

<u>II/ STEP BY STEP</u>

```
.ORIG x3000

LD R6, NEGASCII ;;;; -48

LD R5, NEGASCII9 ;;;; -57

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                                                   ;

;              Input n strings with infitive characters                             ;

;                                                                                   ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

START LEA R0, NHAP_N                                    ; Print string declaring below

PUTS

NHAP_N .STRINGZ "The number of strings (<10) is: "      ; Declare string information for users

AND R2,R2,#0

NHAP_LAI GETC                                           ; Input number of string

        ADD R2,R2,#1

        ADD R4,R0,#-10                                  ; check "Enter" char

        BRz CHECK_ERROR

        ADD R1,R0,R6                                    ; Check number (0-9)?

        BRnz NHAP_LAI

        ADD R1,R0,R5                                    ; Check number (0-9)?

        BRp NHAP_LAI

         ADD R1,R0,R6                                   ; data: number of strings --> R1

        OUT                                             ; Print the number that use input

        BR NHAP_LAI
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                  ;
;                    CHECK ERROR FROM INPUT                        ;
;                                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CHECK_ERROR ST R1, NUMBERSTRING                 ; Check error if first type is not number
ADD R2,R2,#-1
BRp NEXT
LEA R0, ERROR
PUTS                                              ; Print ERROR according to condition
ERROR .STRINGZ " ERROR "                              ; Create string : ERROR
AND R0,R0,#0
ADD R0,R0,#10                                         ; Create newline
OUT
BR START                                              ; Back to input number again if first type is enter
NEXT
AND R0,R0,#0
ADD R0,R0,#10                                         ; Create newline
OUT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                  ;
;                         DECLARE LABEL                            ;
;                                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
BR RELOAD

NUMBERSTRING .BLKW 1

NEG_COMMA .FILL -44

NEG_SPACE .FILL -32

NEGASCII .FILL -48

NEGASCII9 .FILL -57

START_OF_STRING .FILL x5000

;;;
```

```
INPUT GETC                                                    ; Input characters for string

OUT                                                           ; Output character for string

STR R0,R6,#0

ADD R6,R6,#1                              ; Store character to memory of address containing them


ADD R3,R3,#1

ADD R2,R0,#-10                                         ; Indentify "Enter" for starting a new string

BRz CONTROL_STRINGS

BR INPUT

CONTROL_STRINGS ADD R1,R1,#-1              ; Control the number of string to end input character

               BRp INPUT                                  ; Loop of input

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                          ;

;                    RESET ALL REGISTORS                   ;

;                                                          ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R3,R3,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                                                          ;

;                                  JSR OPERATIONS                                          ;

;                                                                                          ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


JSR FILTER_FIRSTCHAR   ; SUBROUTINE to filter the first of character that is blank and comma

JSR RESET_REG                    ; SUBROUTINE to reset registor after filtering the first of character

JSR FILTER_INNERCHAR ; SUBROUTINE to filter the inner character that is redundant blank or
comma


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                                                          ;

;                            RESET ALL REGISTORS                                           ;

;                                                                                          ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R3,R3,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                                     ;
;                  COUNTER CHARACTER OF STRING                        ;
;                                                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LD R6, START_OF_STRING
CONT LDR R0,R6,#0
ADD R2,R2,#1
ADD R3,R3,#1 ;;;; counter tong ky tu
ADD R1,R0,#-10
BRz STORE_COUNTER
BR CONTROL_M
STORE_COUNTER ADD R2,R2,#-1
               AND R2,R2,#0
CONTROL_M ADD R6,R6,#1
LDR R0,R6,#0
BRnp CONT
ADD R3,R3,#-1
SAVECOUNT .BLKW #1
ST R3, SAVECOUNT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                                     ;
;                         RESET ALL REGISTORS                             ;
;                                                                                     ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                             ;

;        FILTER THE LAST CHARACTER (COMMA AND SPACE)          ;

;                                                             ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LD R6,START_OF_STRING                   ; load address of containing character

ADD R6,R6,R3                            ; control address move to the last character

LOOP_1 LDR R0,R6,#0                     ; give value of memory of above address

    ADD R1,R0,#-16

    ADD R1,R1,#-16                      ; check "SPACE"

    BRz ELI_FIRST_1

    ADD R1,R1,#-12                      ; check "COMMA"

    BRz ELI_FIRST_1

    AND R5,R5,#0                        ; counter satisfied occurence is R5

    ADD R5,R5,#1

NORMAL_1

      ADD R0,R0,#-10

      BRz RESET_FOR_ENTER
```

```
   BR CONTROL_1

ELI_FIRST_1 ADD R5,R5,#1

       ADD R5,R5,#-1

       BRz RESET_1                              ; if R5=0 move to RESET_1

       BRp NORMAL_1                      ; if R5=1 move to NORMAL_1 to opeate normally

RESET_FOR_ENTER AND R5,R5,#0

      BR CONTROL_1

RESET_1  STR R2,R6,#0          ; store 0 to memory of address that is space or comma

       AND R5,R5,#0

       BR CONTROL_1

CONTROL_1 ADD R6,R6,#-1

       ADD R3,R3,#-1

       BRzp LOOP_1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                        ;

;                RESET ALL REGISTORS                     ;

;                                                        ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                               ;
;                    JSR OPERATION                              ;
;                                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
JSR FINAL_STRING
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                               ;
;                    OUTPUT FINAL OF STRING                     ;
;                                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
LD R2, NUMBERSTRING
LD R1, ADDR_FIRST_CHAR
ADD R0,R6,#10
OUT
LEA R0, FINAL_RE
PUTS
FINAL_RE .STRINGZ "FINAL DESCENDING STRING: "            ; create string to print on screen
ADD R0,R6,#10
OUT
PRINT_NEXT LDR R0,R1,#0            ; memory of address that contained the address of the first char
PUTS                                                            ; print string
ADD R0,R6,#10
```

```
OUT
ADD R1,R1,#1
ADD R2,R2,#-1
BRnp PRINT_NEXT
HALT
SAVE5 .BLKW 1
ADDR_FIRST_CHAR .FILL x6000
NEG_START .FILL X-5000
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                      ;
;                  FILTER FIRST (COMMA AND SPACE)                      ;
;                                                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FILTER_FIRSTCHAR LD R6,START_OF_STRING                      ; Call address contain character
ADD R4,R6,#0    ; Create temporary variable for above address to store value that is satfistactory
LOOP LDR R0,R6,#0                                 ; Get value from memory of calling above address
   STR R1,R6,#0                                                ; Reset memory
   ADD R3,R0,#-16
   ADD R3,R3,#-16                                              ; Identify "SPACE"
   BRz ELI_FIRST                 ; Move to eliminated first space program according to condition
   ADD R3,R3,#-12                                              ; Identify "COMMA"
   BRz ELI_FIRST                 ; Move to eliminated first comma program according to condition
   AND R5,R5,#0          ; reset variable counter that using to determine character is first or not
   ADD R5,R5,#1
NORMAL
```

```
    STR R0,R4,#0                                        ; Store satisfactory value

    ADD R4,R4,#1                                        ; Increase the address

    ADD R0,R0,#-10                          ; Check "Enter" to reset variable counter "R5"

    BRz RESET

    BR CONTROL

ELI_FIRST ADD R5,R5,#1

        ADD R5,R5,#-1

        BRz RESET     ; If R5=0 --> the first character of string is comma or space --> move to RESET

        BRp NORMAL                                      ; If R5=1 --> next character of string

RESET   AND R5,R5,#0                                    ; Reset counter variable

     STR R5,R6,#0

        BR CONTROL

CONTROL         ADD R6,R6,#1

        LDR R0,R6,#0

        BRnp LOOP

RET                                                     ; return

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                     ;

;               FILTER INNER CHAR OF STRING            ;

;                                                     ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

         ;###########################################################################;

         ;######################## FILTER SPACE ################################;

         ;###########################################################################;

FILTER_INNERCHAR LD R1, NEG_COMMA
```

```
LD R2, NEG_SPACE
LD R6, START_OF_STRING                                  ; Call address containing character
ADD R4,R6,#0                               ; create temporary variable for above address
BACK LDR R0, R6, #0                                     ; get value of address
STR R5,R6,#0                                            ; Store 0 to memory of used address
ADD R3,R0,R2                                            ; Check "SPACE"
BRnp NOTSPACE                                           ; Move to NOTSPACE if not space
STR R0,R4,#0                                            ; Store satisfactory value
ADD R4,R4,#1                                            ; Increase satisfactory address
NEXT_CHAR ADD R6,R6,#1                                  ; Check next char
        LDR R0,R6,#0                                    ; get value of address
        STR R5,R6,#0                                    ; Store 0 to memory of used address
        ADD R3,R0,R2                                    ; Check "SPACE"

        BRz NEXT_CHAR                                   ; Condition if char is space
NOTSPACE
    STR R0,R4,#0                                        ; Store satisfactory value
       ADD R4,R4,#1                                     ; Increase satisfactory address
       ADD R6,R6,#1                              ; Increase old address to check next char
       LDR R0,R6,#0
       BRnp BACK
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                      ;
;                 RESET ALL REGISTORS                  ;
;                                                      ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R3,R3,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0

            ;###########################################################################;

            ;########################## FILTER COMMA ##################################;

            ;###########################################################################;

LD R1, NEG_COMMA

LD R6, START_OF_STRING                                        ; Call address containing character

ADD R4,R6,#0                                ; create temporary variable for above address

BACK_1 LDR R0, R6, #0                                         ; get value of address

STR R5,R6,#0                                                  ; Store 0 to memory of used address

ADD R3,R0,R1                                                  ; Check "COMMA"

BRnp NOTCOMMA                                         ; Move to NOTCOMMA if not comma

STR R0,R4,#0                                                  ; Store satisfactory value

ADD R4,R4,#1                                                  ; Increase satisfactory address

ADD R0,R0,#-10

NEXT_CHAR_1 ADD R6,R6,#1                                      ; Check next char

          LDR R0,R6,#0                                        ; get value of address

          STR R5,R6,#0                                        ; Store 0 to memory of used address

          ADD R3,R0,R1                                        ; Check "COMMA"
```

```
            BRz NEXT_CHAR_1                                    ; Condition if char is comma

NOTCOMMA STR R0,R4,#0                                          ; Store satisfactory value

           ADD R4,R4,#1                                        ; Increase satisfactory address

           ADD R6,R6,#1                                ; Increase old address to check next char

           LDR R0,R6,#0

           BRnp BACK_1

RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                          ;

;                   STORE FINAL STRING IN DATA             ;

;                                                          ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FINAL_STRING LD R3,SAVECOUNT                                   ; load the total character of string

LD R6, START_OF_STRING                                         ; load address store each char

ADD R4,R6,#0                                   ; create variable temporary of above address

BACK_FINAL LDR R0,R6,#0                                ; load memory of that above address

            STR R1,R6,#0                               ; Store 0 in the memory of the used address

   BRz CONTROL_FINAL

   ADD R2,R0,#-10                                              ; Check "ENTER"

   BRz STORE_0                                         ; condition to stor O if character is "ENTER"

   STR R0,R4,#0       ; store satisfactory value to memory of address that is initalized as temporary

   ADD R4,R4,#1                                                ; increase satisfied address

   ADD R5,R5,#1

   BR CONTROL_FINAL
```

```
STORE_0 STR R2,R4,#0                    ; Store 0 to memory of address that contained "ENTER"

        ADD R4,R4,#1                                          ; Increase satisfied address

        ADD R5,R5,#1

CONTROL_FINAL

        ADD R6,R6,#1

        ADD R3,R3,#-1

        BRzp BACK_FINAL

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                    ;

;                  DECLARE LABLE                     ;

;                                                    ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

NUMBERKYTU .BLKW #1

ST R5,NUMBERKYTU

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                    ;

;       STORE ADDRESS OF FIRST CHAR OF EACH STRING    ;

;                                                    ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; TONG CAC KY TU --> R5 = 31

LD R6, START_OF_STRING

LD R5, ADDR_FIRST_CHAR

LD R2, NUMBERSTRING

STR R6,R5,#0

TIEP LDR R0,R6,#0
```

```
ADD R6,R6,#1

ADD R1,R0,#0

BRz STOREADDR

BRnp TIEP

STOREADDR ADD R2,R2,#-1

              BRz SORT

              ADD R5,R5,#1

              STR R6,R5,#0

BR TIEP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                                                                  ;

;                              RESET ALL REGISTORS                                                 ;

;                                                                                                  ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                                                                  ;

;                                      SORT STRING                                                 ;

;                                                                                                  ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;; NOTE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;HAVE: NUMBERSTRING, ADDR_FIRST_CHAR (6000->), NUMBERKYTU;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SORT     ADD R0,R0,#0

;########################################################################################;

MAX             LD  R4, NUMBERSTRING            ; load number of strings

OUTER_1         ADD R4, R4, #-1         ; loop n - 1 times

                BRnz NEXT_7             ; looping complete, exit

                ADD R6, R4, #0          ; initialize inner loop counter to outer

                LD  R1, ADDR_FIRST_CHAR ; set file pointer to begin of file

INNER_1         LDR R2, R1, #0          ; load address contain address of first letter of strings

                LDR R0, R2, #0          ; load the first letter

                LDR R3, R1, #1   ; load address contain address of first letter of the next strings

                LDR R5, R3, #0          ; load the first letter of the next strings

                ST  R5, SAVE5

                NOT R5, R5              ; compare two first letter

                ADD R5, R5, #1

                ADD R5, R0, R5

                BRp SWAP_1              ; if the result negative change the address of two first letter

                LD  R5, SAVE5

                STR R3, R1, #0

                STR R2, R1, #1

SWAP_1          LD R5, SAVE5

                ADD R1, R1, #1          ; increment file pointer
```

```
                ADD R6, R6, #-1         ; decrement inner loop counter

                BRp INNER_1             ; end of inner loop

                BR  OUTER_1             ; end of outer loop

NEXT_7          RET

;##############################################################################################################
#####;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;                                                          ;

;                    RESET ALL REGISTORS                    ;

;                                                          ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0

RET

RESET_REG AND R0,R0,#0

AND R1,R1,#0

AND R2,R2,#0

AND R3,R3,#0

AND R4,R4,#0

AND R5,R5,#0

AND R6,R6,#0
```

RET

.END

III/ OPERATE CODE

1. Input number from keyboard to input the number of strings. Press Enter to start input character of string.
2. Input character from keyboard and put **ENTER** from keyboard to finish each string.
3. If you entered enough the number strings being equivalent to your requirement in the first step, you will see the final result in the term “FINAL DESCENDING STRING: “.
4. If you want to try another times you want to reinitialize machine to do it

IV/ SAMPLE RESULT