VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

COMPUTER SYSTEMS AND PROGRAMMING (EE2415)
CLASS TT01 - SEMESTER 222

PROJECT LC3

Student: Luong Hoang Phuc 2010525

HO CHI MINH CITY, MAY 2023

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.

```
.orig x3000

instructno  .stringz "Please inform the number of inputing strings: "
instructinput   .stringz "Now, please enter you strings gradually: \n NOTED THAT
ENTERED NUMBERS MUST HAVE THE SAME DIGITS \N(EX: 001, 563, 072, 693)"
instructend     .stringz "Double ENTER if you want to end input"
displaying  .stringz "The sorted array of strings are:"

;reset all register

    jsr     reset
;                                                                       (1)
;input strings with r0 --> storing input
;                   r1 --> status register
;                   r2 --> number of strings
;                   r3 --> pointer of character

    lea     r0, instructno
    puts
;   input a number

    lea     r5, numberchar

    numin
    ldi     r1, kbsr    ; if ready to input number
    brzp    numin
    ldi     r0,kbdr

    and     r6,r6,#0
    ld  r4,n_number
    add     r4,r4,r0
    jsr     no_range    ; with variable r4=r0=check=indicator
    add     r6,r6,#0 ; recall r6
    brz     numend ; if r6=0, then end inputing tasks

    numout
    ldi     r1,dsr
    brzp    numout ; if ready to out number on screen
    sti     r0,ddr
    ld  r4,n_number
    add     r0,r4,r0    ; convert to semi-decimal value from ASCII
    str     r0,r5,#0    ; store the value to memory[r5]

    add     r5,r5,#1    ; increase r5 to next location storage
    add     r3,r3,#1 ; increase index weight to convert to decimal value latter

    br  numin
;   End input number
;                                                           (2)
    numend

    add     r3,r3,#-1   ; just decrease r3 to make enough
    st  r3,save3    ; Store offset
    add     r5,r3,#0 ; if index weight =0, dont need to convert
    brz     addlocate
```

```
; Varable r0=ouput , r1<->r2= variable
; Result r6
    lea  r5,numberchar ; reload locate store the input number
    and     r6,r6,#0 ; the decimal value will be outcome of r6
;
    readd
    and     r1,r1,#0
    add     r1,r1,#1 ; preset r1
    and     r0,r0,#0
    add     r4,r3,#0 ; store index weight to r4
    st  r1,temp ; store 1 to temporary memory

;calculation 10 power nth

    repower
    and     r2,r2,#0
    add     r2,r2,#10 ; from the sbr c=a*10
    ld  r1,temp ; the starting value c=1*10; then c=temp*10
    jsr     mutpl ;
    st  r0,temp ;store the output 10^(n-1) to temporary memory
    add     r4,r4,#-1
    brp     repower  ; if index =0, we get the exact weight[r4]=temp
;calculation weight

    add     r3,r3,#0 ; recall r3

    ld  r1,temp ; load weight of index to r1
    ldr     r2, r5,#0 ; load the coefficent of the weight that previously store
to mem[r5]
    jsr     mutpl
    add     r6,r6,r0 ;accumulate the value of coef*weight[n]=r6

    add     r5,r5,#1 ; increase to next coefficient
    add     r3,r3,#-1 ; decrease index from n to zero
    brp readd

    addlocate   ; the phase tasks to accumulate the final unit value
coef*weight[0]
    lea     r5,numberchar ;reobtain to location with semi-decimal value
    ld  r3,save3 ; restore offset
    add     r5,r5,r3 ; mem[r5+r3] is actually the unit value
    ldr     r0,r5,#0
    add     r6,r6,r0 ; accumulate the unit value
    st r6,number   ; Store practical value to location "number"
;                                                       (3)
;
    jsr     reset
; print instruction
    add     r0,r0,#10
    out
    out
    lea     r0, instructend
    puts
    add     r0,r1,#10
    out
    lea     r0, instructinput
    puts
    add r0,r1,#10
    out
;input stringz --> r3 = pointer, r5 =condition, r6 =location, r4 =saving
    ld  r3,number   ; r3 is the number of inputing strings you required
    ld  r4,strings  ;location starting to store strings is mem[x4080]
    ld  r2,start    ;data x4080 to store the a memory file of starting adress
; store x4080 = mem[x4000], as it is obvious
;this task is just aimed to preset
```

```
    ld  r1, charptr
    add     r1,r1,#-1
    str     r2,r1,#0
    ld  r1, charptr
    st  r1,temp ;store charater pointer to temporary memory
;reset r2,ready to input
    and     r2,r2,#0

    str_in
    ldi     r1, kbsr
    brzp    str_in ; is it ready to input
    ldi     r0,kbdr

    str_out
    ldi     r1,dsr
    brzp    str_out ; is it ready to output
    sti     r0,ddr

    str     r0,r4,#0 ; store inputing strings to r0=mem[r4++]
    add     r4,r4,#1 ; increase the character memory pointer


    add     r5,r0,#-10 ; -10=-xA = newline
    brnp    str_in ;If input = newline, then move to store value

    add     r2,r2,#1 ;once i met an enter, then it is a strings
;This r2 value use for latter purposes
    ld  r1,temp ; get the location of memory file
    str     r4,r1,#0
    add     r1,r1,#1 ; increase loaction
    st  r1,temp ; store to temporary memory

    jsr     check ; go to check wheter the next char is #-10
    jmp     r6 ;this jump to str_out or str_end

;   End input strings then load to memory
;
    str_end
    st  r2,real_no ; store the real number of strings
;                                               (4)
;    Choose way to sort ; Informing users

    jsr     reset
    add     r0,r0,#10
    out
    lea     r0,sortinga
    puts
    add     r0,r1,#10
    out
    lea     r0,sortingd
    puts

    rein
    add     r0,r1,#10
    out
    in  ; input just one word
    jsr     checksort ; check wheter character is acceptable
    jmp     r6 ; this jump to rein or next instruction
    add     r0,r0,#0 ; recall r0
    brz     descendsort ; if r0=0 then des-sort, otherwise as-sort
    jsr     ascending ; this tasks sort the strings
    br  endsorting

    descendsort
```

```
    jsr     descending ; this tasks sort the strings
    endsorting
;                                                                   (5)
;Displaying Results

    jsr     reset
;Displaying prompts
    add     r0,r0,#10
    out
    lea     r0,displaying
    puts
    add     r0,r1,#10
    out
;
    jsr     reset
;
;Preset variable
    ld  r6,startadd     ;take the location where saving the starting address
    ld  r2,real_no  ;take the number of strings
    add     r2,r2,#1    ; plus 1 to ensure sufficient tasks
    and     r4,r4,#0    ;preset r4

    nextstr_out
    st  r6,save6
    ldr     r5,r6,#0 ; take starting address
    add     r2,r2,#-1 ; pointer decrement
    brz     end_str_out ; if it is 0, then we finish printing strings
    and r6,r6,#0 ;preset r6

    reprint
    ldr     r0,r5,#0 ;get the character in the string
    jsr     rbd_out ; remove rebudant comma and space

;in which r0 is output
;       r5 is location memory
    add     r5,r5,#1
    add     r1,r0,#-10 ;check whether it met an enter
    brnp    reprint

    ld  r6,save6
    add     r6,r6,#1    ;move to next starting address
    br  nextstr_out

end_str_out


halt

;                                                                   (6)
;   Display prompt
    sortinga    .stringz "Type a or A for ascending sort"
    sortingd    .stringz "Type d or D for descending sort"
;   ASCII code
n_number .fill x-30
newline .fill x0a
space   .fill x20
colon   .fill x2c
n_space .fill x-20
n_comma .fill x-2c

;   save location
save0 .blkw #1
save1 .blkw #1
save2 .blkw #1
save3 .blkw #1
```

```
save4 .blkw #1
save5 .blkw #1
save6 .blkw #1
save7 .blkw #1
temp .blkw #1
;   status and data location
kbsr .fill xfe00
kbdr .fill xfe02
dsr .fill xfe04
ddr .fill xfe06
;     number store location
number .blkw #1
numberchar .blkw #6
;   strings saving
strings .fill x4080
;ascii char
a .fill #-97
A .fill #-65
d .fill #-100
D .fill #-68
; character pointer
start .fill x4080
charptr .fill x4001
startadd .fill x4000
real_no .blkw #1
no_temp .blkw #1
;                                                                    (7)
;                                                                     (a)
;   reset all register
reset
    st  r7,save7
    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
    ld  r7,save7
    ret
;                                                                    (b)
;   check in range number


no_range
    st  r7,save7
    st  r5,save5 ; store all value existing in r5,r7
    brn     endno_range ; if the former input r0 <0, meaning that not in number
range , just end the inputing number tasks
    add     r4,r4,#-9 ; if the former input r0>9, not in range, just end
    brp     endno_range
    add     r6,r6,#1  ; the signal r6 =1 means that in number range, 0 means no
to check latter

endno_range

    ld  r7,save7
    ld  r5,save5 ; restore the value
    ret
;                                                                (c)
;   Multiplcation with c=a*b
;                   in which c=r0
;                            a=r1
;                            b=r2
    mutpl
```

```
    st  r7,save7

    and     r0,r0,#0 ; reset the outcome

    startmutpl
    add     r0,r0,r1 ; gradually add r1 to the outcome
    add     r2,r2,#-1 ; decrease the pointer which is also b
    brp     startmutpl ; until it gets 0, then end the mutiply tasks

    ld  r7,save7
    ret
;                                                               (d)
;
;   Check wheter it is an end signal
;   With r6= jump location
;           r5 = condition

check
    st  r7,save7
    add     r3,r3,#-1 ; if r3=number of former informing number is out, then just
end input strings
    brz  strend_ch

str_inch
    ldi     r1, kbsr ; continue to get the next character input
    brzp    str_inch
    ldi     r0,kbdr
    add     r5,r0,#-10
    brnp    check_next ; if input is not a feedline, then just go to print out on
screen

strend_ch
    lea     r6,str_end ; store the location to jump to end inputing
    br  check_end ; then just go to end sbr

check_next
    lea     r6,str_out ;store the location where going to print character on
screen to jump latter

check_end
    ld  r7,save7 ; get PC jump location
    ret
;                                                                   (e)
;   check for sort --> output r0=1 for a or r0=0 for d, r5 check
checksort
    st  r7,save7 ; store jump PC location
    ld  r6,a ; get -a
    add     r5,r0,r6
    brz ascend ; if input r0=-a r6, acceptable to ascend

    ld  r6,A ; get -A
    add     r5,r0,r6
    brz     ascend ; if input r0= -A r6, acceptable to ascend

    ld  r6,d ; get -d
    add     r5,r6,r0
    brz     descend ; if input r0 = -d r6, acceptable to descend

    ld  r6,D ; get -D
    add     r5,r6,r0
    brz     descend ; if input r0 = -D r6, acceptable to descend

    lea     r6,rein ; if no acceptable value, get address of reinputing phrase to
jump latter
    br  endsort ; then end checking
```

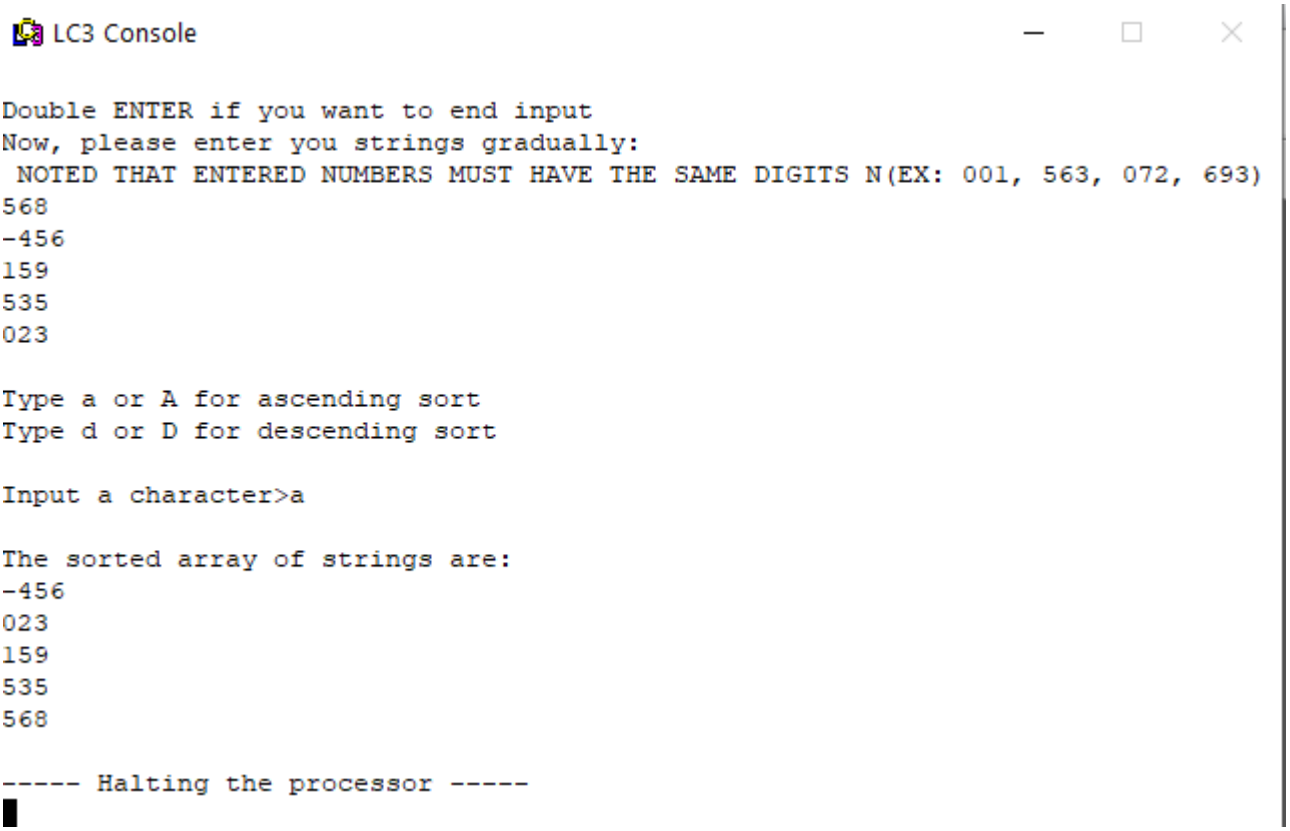
```
ascend
    and     r0,r0,#0
    add     r0,r0,#1 ; if ascend, the signal =1 to check when out sbr
    ld  r6,save7
    add     r6,r6,#1 ;store address r6 to jump when out sbr
    br  endsort

descend
    and     r0,r0,#0 ; if descend, the signal=0 to check when out sbr
    ld  r6,save7
    add     r6,r6,#1 ; store address r6 to jump when out sbr

endsort
    ld  r7,save7 ; get the jump PC address
    ret
;                                                                   (f)
;
;   Ascending sort
ascending
    st  r7,save7
    ld  r0,startadd ; get the loaction storing the start adress of each strings
    st  r0,save0 ; store the value to temporary memory

    and     r5,r5,#0 ; --> r5=i
    and     r6,r6,#0 ; --> r6=j


loop_i
    ld  r2,real_no
    not     r2,r2
    add     r2,r2,#2 ; obtain -(r2-1)
    add     r0,r5,r2
    brp     end_i ;if i>number of string, then end loop i
    add r6,r5,#1

loop_j
    ld  r2,real_no
    not     r2,r2
    add     r2,r2,#2 ; obtain -(r2-1)
    add     r0,r6,r2
    brp     end_j ; if j>number of string, then end loop j

    ld  r0,save0
    add     r0,r0,r5
    ldr     r3,r0,#0  ; r3 keep start location index i
    ld  r0,save0
    add     r0,r0,r6
    ldr     r4,r0,#0 ; r4 keep start location index j

check_ascend
    ldr     r1,r3,#0 ; first char of string[i]
    add     r0,r1,#-10
    brz     replace_end


    ldr     r2,r4,#0 ; first char of string[j]
    add     r0,r2,#-10
    brz     replace
; get 2's com r2 = -r2
    not     r2,r2
    add     r2,r2,#1
;
    add     r0,r1,r2
    brn     replace_end ; if r1<r2, then just increase pointer
```

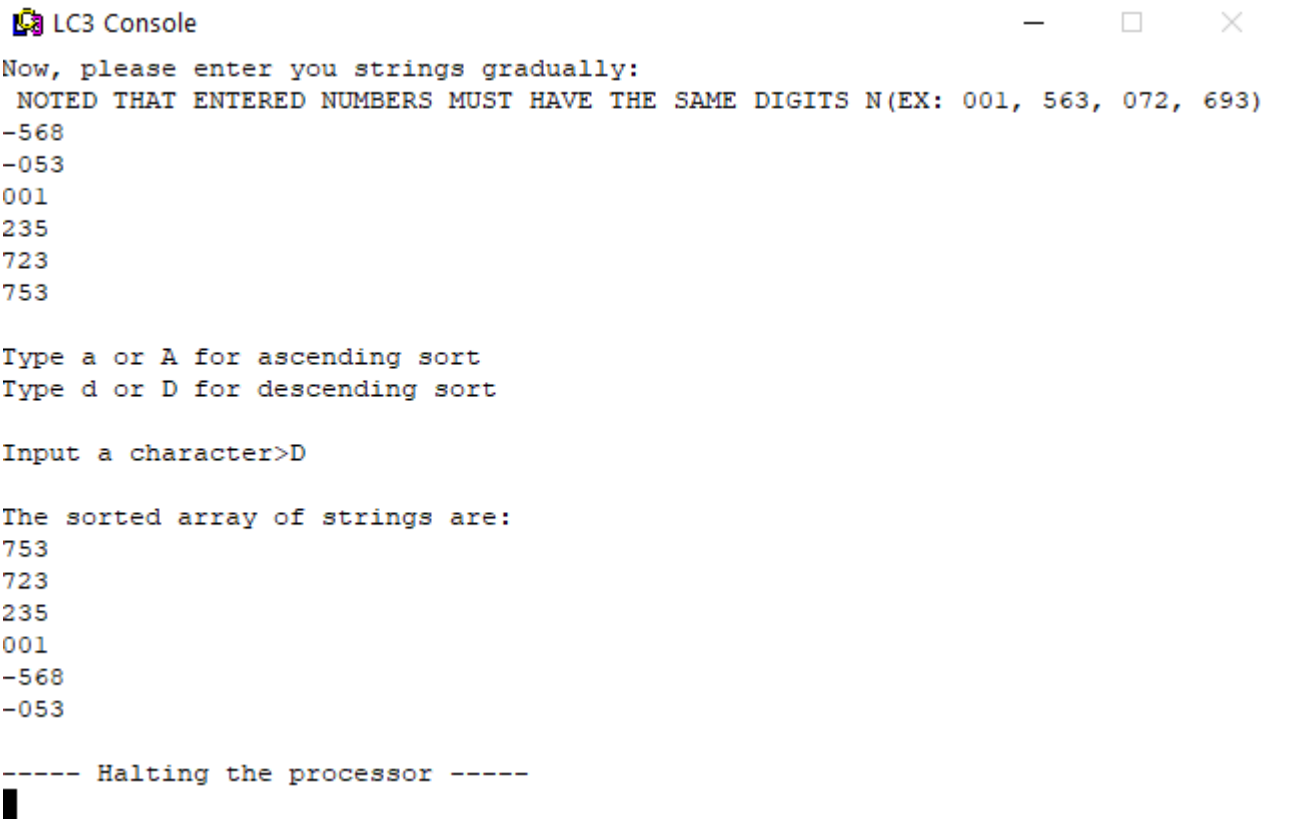
```
    brp     replace

    add     r3,r3,#1
    add  r4,r4,#1
    br  check_ascend

replace
    ld  r0,save0
    add     r0,r0,r5
    ldr     r1,r0,#0 ; --> r1=[i]

    ld  r0,save0
    add     r0,r0,r6
    ldr     r2,r0,#0; --> r2=[j]


    ld  r0,save0
    add     r0,r0,r5
    str  r2,r0,#0 ; store mem[j] to mem[i]

    ld  r0,save0
    add     r0,r0,r6
    str     r1,r0,#0 ; store mem[i] to mem[j]


replace_end
    add     r6,r6,#1 ; increase r6=j
    br  loop_j ; reloop until r6=j met condition
end_j
    add     r5,r5,#1 ; increase r6 = i
    br  loop_i ; reloop until r5=i met condition
end_i
    ld  r7,save7 ; resgain location r7
    ret

;                                                                       (g)
;
;   Descending sort
descending
    st  r7,save7
    ld  r0,startadd ; get the loaction storing the start adress of each strings
    st  r0,save0 ; store the value to temporary memory

    and     r5,r5,#0 ; --> r5=i
    and     r6,r6,#0 ; --> r6=j


loop_id
    ld  r2,real_no
    not     r2,r2
    add     r2,r2,#2 ; obtain -(r2-1)
    add     r0,r5,r2
    brp     end_id ;if i>number of string, then end loop i
;
    add     r6,r5,#1

loop_jd
    ld  r2,real_no
    not     r2,r2
    add     r2,r2,#2 ; obtain -(r2-1)
    add     r0,r6,r2
    brp     end_jd ; if j>number of string, then end loop j

    ld  r0,save0
    add     r0,r0,r5
```

```
    ldr     r3,r0,#0  ; r3 keep start location index i
    ld  r0,save0
    add     r0,r0,r6
    ldr     r4,r0,#0 ; r4 keep start location index j

check_descend
    ldr     r1,r3,#0 ; first char of string[i]
    add     r0,r1,#-10
    brz     replaced


    ldr     r2,r4,#0 ; first char of string[j]
    add     r0,r2,#-10
    brz     replace_endd
; get 2's com r2 = -r2
    not     r2,r2
    add     r2,r2,#1
;
    add     r0,r1,r2
    brp     replace_endd ; if r1<r2, then just increase pointer
    brn     replaced

    add     r3,r3,#1
    add     r4,r4,#1
    br  check_descend

replaced
    ld  r0,save0
    add     r0,r0,r5
    ldr     r1,r0,#0 ; --> r1=[i]

    ld  r0,save0
    add     r0,r0,r6
    ldr     r2,r0,#0; --> r2=[j]


    ld  r0,save0
    add     r0,r0,r5
    str     r2,r0,#0 ; store mem[j] to mem[i]

    ld  r0,save0
    add     r0,r0,r6
    str     r1,r0,#0 ; store mem[i] to mem[j]


replace_endd
    add     r6,r6,#1 ; increase r6=j
    br  loop_jd ; reloop until r6=j met condition
end_jd
    add     r5,r5,#1 ; increase r6 = i
    br  loop_id ; reloop until r5=i met condition
end_id
    ld r7,save7 ; resgain location r7
    ret
;                                                                   (h)
;  Remove rebudant spacebar and colon
;  In which r0 is output
;           r5 is location memory
rbd_out
    st  r7,save7 ; store jump PC address

    br  rbd_space
rbd_sp_out
    add     r4,r4,#0 ; all r4 was preseted at 0 unless a spacebar reoccur
    brp     rbd_space ; once r4=1, we then cannot printout the character
```

```
    out
    add     r4,r4,#1 ; the spacbar may reoccur
rbd_space
    ldr     r0,r5,#0 ; get the next char to r0
    ld  r3,n_space ; r3=-32
    add     r1,r0,r3
    brnp    rbd_comma ;if r0 is not a spacebar, check whether it is a colon
    add     r5,r5,#1 ; if yes, increase to next char in strings
    br  rbd_sp_out ; check wheter it reoccur not to print out

rbd_co_out
    add     r6,r6,#0 ;all r6 was preseted at 0 unless a comma reoccur
    brp     rbd_comma ;once r6=1, we then cannot printout the character
    out
    add     r6,r6,#1 ;the comma may reoccur
rbd_comma
    ldr     r0,r5,#0 ; get next character to r0
    ld  r3,n_space
    add     r1,r0,r3
    brz     rbd_space ;check whether a spacebar appear after comma
    ld  r3,n_comma
    add     r1,r0,r3
    brnp    rbd_end ; if it is the comma, then end tasks and output char
    add     r5,r5,#1 ; increase to next character location
    br  rbd_co_out ; check whether it reoccur not to print out

rbd_end
out
    and     r4,r4,#0 ; reset r4
    and     r6,r6,#0 ; reset r6
    ld  r7,save7 ; back to the main program
    ret
.end
```

SIMULATION


LC3 Console
Double ENTER if you want to end input
Now, please enter you strings gradually:
NOTED THAT ENTERED NUMBERS MUST HAVE THE SAME DIGITS N(EX: 001, 563, 072, 693)
568
-456
159
535
023
Type a or A for ascending sort
Type d or D for descending sort
Input a character>a
The sorted array of strings are:
-456
023
159
535
568
----- Halting the processor -----


LC3 Console
Now, please enter you strings gradually:
NOTED THAT ENTERED NUMBERS MUST HAVE THE SAME DIGITS N(EX: 001, 563, 072, 693)
-568
-053
001
235
723
753
Type a or A for ascending sort
Type d or D for descending sort
Input a character>D
The sorted array of strings are:
753
723
235
001
-568
-053
----- Halting the processor -----