```
%include "asm_io.inc"

SECTION .data

msg1: db "incorrect number of command line arguments",0
msg2: db "inccorect length of the argument",0
msg3: db "inccorect first letter of the argument (should be 3 or 5 or 7 or 9)",0
msg4: db "inccorect second letter of the argument (should be an upper case letter)",0
msg5a: db "Displaying shape of size ",0
msg5b: db " made of letters ",0

SECTION .bss
SIZE: resd 1
LETTER: resb 1

SECTION .text
   global  asm_main

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
asm_main:
   enter 0,0             ; setup routine

   ;; check argc, must be 2
   mov eax,[ebp+8]  ;; argc
   cmp eax,2
   je Check_length  ;; check length of argv[1]
   mov eax,msg1     ;; display argc error message
   call print_string
   call print_nl
   jmp asm_main_end

Check_length:
   mov ebx,[ebp+12] ;; address of argv[]
   mov ecx, [ebx+4] ;; address of argv[1]
   ;; first letter should not be NULL
   cmp byte [ecx],byte 0
   je Bad_length
   ;; the second letter should not be NULL
   cmp byte [ecx+1], byte 0
   je Bad_length
   ;; the third letter should be NULL
   cmp byte [ecx+2], byte 0
   jne Bad_length
   jmp Length_ok

Bad_length:
   mov eax, msg2
   call print_string
   call print_nl
   jmp asm_main_end

Length_ok:
   cmp byte[ecx],'0'
   jb Bad_first_letter
   cmp byte[ecx],'9'
   ja Bad_first_letter
   cmp byte[ecx],'3'
   je Set_size
   cmp byte[ecx],'5'
   je Set_size
   cmp byte[ecx],'7'
   je Set_size
   cmp byte[ecx],'9'
   je Set_size
   jmp Bad_first_letter

Set_size:
   mov al, byte [ecx]
   sub eax, dword '0'
   mov [SIZE], eax
   jmp Check_second_letter

Bad_first_letter:
   mov eax,msg3
   call print_string
   call print_nl
   jmp asm_main_end

Check_second_letter:
   inc ecx;            ;; address of the second letter
   cmp byte[ecx],'A'
   jb Bad_second_letter
   cmp byte[ecx],'Z'
   ja Bad_second_letter
   ;; store letter in LETTER
   mov al, byte [ecx]
   mov [LETTER], al
   jmp  Arg_ok

Bad_second_letter:
   mov eax, msg4
   call print_string
   call print_nl
   jmp asm_main_end

Arg_ok:
   mov eax, msg5a
   call print_string
   mov eax, [SIZE]
   call print_int
   mov eax, msg5b
   call print_string
   mov al, [LETTER]
   call print_char
   call print_nl

 asm_main_end:
   leave
   ret
```