```
%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

SECTION .bss
SIZE: resd 1
LETTER: resb 1

SECTION .text
   global  asm_main

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; subroutine display_line
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
display_line:
   enter 0,0             ; setup routine
   pusha                 ; save all registers

   mov ebx, [ebp+8]      ; bl is the letter
   mov ecx, [ebp+12]     ; number of letters
   mov edx, [ebp+16]     ; number of spaces

   mov esi, 1
   L1: cmp esi, edx
       ja L2
       mov al,' '
       call print_char
       inc esi
       jmp L1
   L2: mov esi, 1
   L3: cmp esi, ecx
       ja L4
       mov al, bl
       call print_char
       mov al, ' '
       call print_char
       inc esi
       jmp L3
   L4: call print_nl
   popa
   leave
   ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; subroutine display_line ends
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;




;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; subroutine display_shape
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
display_shape:
   enter 0,0             ; setup routine
   pusha                 ; save all registers

   mov ebx, [ebp+8]      ;bl is the letter
   mov ecx, [ebp+12]     ;ecx is the size
```

```
    cmp ecx, dword 3
    jne D1
    mov eax, dword 2
    jmp D4

    D1: cmp ecx, dword 5
    jne D2
    mov eax, dword 4
    jmp D4

    D2: cmp ecx, dword 7
    jne D3
    mov eax, dword 6
    jmp D4

    D3:
    mov eax, dword 8

    D4:
    ;; bl is the letter
    ;; ecx is the size
    ;; eax is the midpoint
    ;; edx controls the loop
    mov edx, dword 1
    D5: cmp edx, ecx
    jae D6
       push eax      ;; the number of spaces
       push dword 1  ;; the number of letters
       push ebx  ;; the letter
       call display_line
       add esp, 12
       inc edx
       jmp D5
    D6:
       push dword 0  ;; the number of spaces
       push ecx      ;; the number of letters
       push ebx      ;; the letter
       call display_line
       add esp, 12

 display_shape_end:
    popa
    leave
    ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; subroutine display_shape end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;




;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; subroutine 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:
    push dword [SIZE]
    mov eax, 0
    mov al, byte [LETTER]
    push eax
    call display_shape
    add esp, 8

  asm_main_end:
    leave
    ret
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; subroutine asm_main end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```