```
INCLUDE Irvine32.inc
INCLUDE Macros.inc
INCLUDE VirtualKssSeys.inc

.data

endl EQU <0dh,0ah>   ; end of line sequence
message LABEL BYTE
        BYTE "                                                  " ,endl
        BYTE "                                                  " ,endl
        BYTE "    __  __ _______ __   _ _______ __  __ _______ __   _     " ,endl
        BYTE "   | | | ||  _  || | | | ||     || |_| ||  _  || | | |    " ,endl
        BYTE "   | |_| || |_| ||  |_| ||  ___||     || |_| ||  |_| |    " ,endl
        BYTE "   |     ||     ||     ||  | __ |     ||     ||     |    " ,endl
        BYTE "   |     ||     || _   ||  ||  ||     ||     || _   |    " ,endl
        BYTE "   |  _  ||  _  || | |  ||  |_| || ||_|| ||  _  || | |  |    " ,endl
        BYTE "   |__| |__||__| |__||_|  |__||_______||_|  |_||__| |__||_|  |__|    " ,endl
        BYTE "                                                  " ,endl
        BYTE "   Copyright (C) 2021  Mohsin Ali Mirza    20K0305                " ,endl
        BYTE "                  Syed Muhammad Faheem 20K1054                  " ,endl
        BYTE "                  Ahmad Aleem         20K0169                 " ,endl
        BYTE "                                                  " ,endl
messageSize DWORD ($-message)
consoleHandle HANDLE 0    ; handle to standard output device
bytesWritten  DWORD ?     ; number of bytes written

HANGMAN_GOODGAME_00 LABEL BYTE
                                        BYTE "+------+      ",endl
```

```
        BYTE "|      |      ",endl
        BYTE "|            ",endl
        BYTE "|      O      ",endl
        BYTE "|     /|\     ",endl
        BYTE "|     / \     ",endl
        BYTE "+-------------+ ",endl
        BYTE "| YOU   WIN  | ",endl
        BYTE "+-------------+ ",endl
messageSizeGoodGame DWORD ($-HANGMAN_GOODGAME_00)

HANGMAN_GOODGAME_01 LABEL BYTE
                                BYTE "+------+      ",endl
        BYTE "|      |      ",endl
        BYTE "|            ",endl
        BYTE "|      O_     ",endl
        BYTE "|     /|      ",endl
        BYTE "|     / \     ",endl
        BYTE "+-------------+ ",endl
        BYTE "| YOU   WIN  | ",endl
        BYTE "+-------------+ ",endl

HANGMAN_GOODGAME_02 LABEL BYTE
                                BYTE "+------+      ",endl
        BYTE "|      |      ",endl
        BYTE "|            ",endl
        BYTE "|      O/     ",endl
        BYTE "|     /|      ",endl
        BYTE "|     / \     ",endl
```

```
        BYTE "+------------+ ",endl
        BYTE "| YOU   WIN  | ",endl
        BYTE "+------------+ ",endl

HANGMAN_GOODGAME_03 LABEL BYTE
                                BYTE "+------+      ",endl
        BYTE "|      |      ",endl
        BYTE "|            ",endl
        BYTE "|      O_     ",endl
        BYTE "|     /|      ",endl
        BYTE "|     / \     ",endl
        BYTE "+------------+ ",endl
        BYTE "| YOU   WIN  | ",endl
        BYTE "+------------+ ",endl

HANGMAN_GAMEOVER_00 LABEL BYTE
                                BYTE "+------+      ",endl
        BYTE "|      |      ",endl
        BYTE "|      O      ",endl
        BYTE "|     /|\     ",endl
        BYTE "|     / \     ",endl
        BYTE "|            ",endl
        BYTE "+------------+ ",endl
        BYTE "|  YOU  DIE  | ",endl
        BYTE "+------------+ ",endl

HANGMAN_GAMEOVER_01 LABEL BYTE
                                BYTE "+------+      ",endl
```

```
        BYTE "|     /       ",endl
        BYTE "|   _O        ",endl
        BYTE "|  _/\        ",endl
        BYTE "|   \         ",endl
        BYTE "|             ",endl
        BYTE "+-------------+ ",endl
        BYTE "|  YOU  DIE   | ",endl
        BYTE "+-------------+ ",endl

HANGMAN_GAMEOVER_02 LABEL BYTE
                                BYTE "+------+      ",endl
        BYTE "|      |      ",endl
        BYTE "|      O      ",endl
        BYTE "|     /|\     ",endl
        BYTE "|     / \     ",endl
        BYTE "|             ",endl
        BYTE "+-------------+ ",endl
        BYTE "|  YOU  DIE   | ",endl
        BYTE "+-------------+ ",endl

HANGMAN_GAMEOVER_03 LABEL BYTE
                                BYTE "+------+      ",endl
        BYTE "|      \      ",endl
        BYTE "|       O_    ",endl
        BYTE "|       /\_   ",endl
        BYTE "|        /    ",endl
        BYTE "|             ",endl
        BYTE "+-------------+ ",endl
```

```
        BYTE "|  YOU  DIE  | ",endl
        BYTE "+------------+ ",endl

HANGMAN_LIVES_06 LABEL BYTE
                                BYTE "+------+      ",endl
                                BYTE "|      |      ",endl
                                BYTE "|             ",endl
                                BYTE "|             ",endl
                                BYTE "|             ",endl
                                BYTE "|             ",endl
                                BYTE "+------------+ ",endl
                                BYTE "|           | ",endl
                                BYTE "+------------+ ",endl

HANGMAN_LIVES_05 LABEL BYTE
                                BYTE "+------+      ",endl
                                BYTE "|      |      ",endl
                                BYTE "|      O      ",endl
                                BYTE "|             ",endl
                                BYTE "|             ",endl
                                BYTE "|             ",endl
                                BYTE "+------------+ ",endl
                                BYTE "|           | ",endl
                                BYTE "+------------+ ",endl

HANGMAN_LIVES_04 LABEL BYTE
                                BYTE "+------+      ",endl
                                BYTE "|      |      ",endl
```

```
                    BYTE "|      O      ",endl
                    BYTE "|      |      ",endl
                    BYTE "|             ",endl
                    BYTE "|             ",endl
                    BYTE "+-------------+ ",endl
                    BYTE "|           | ",endl
                    BYTE "+-------------+ ",endl

HANGMAN_LIVES_03 LABEL BYTE
                    BYTE "+------+      ",endl
                    BYTE "|      |      ",endl
                    BYTE "|      O      ",endl
                    BYTE "|     /|      ",endl
                    BYTE "|             ",endl
                    BYTE "|             ",endl
                    BYTE "+-------------+ ",endl
                    BYTE "|           | ",endl
                    BYTE "+-------------+ ",endl

HANGMAN_LIVES_02 LABEL BYTE
                    BYTE "+------+      ",endl
                    BYTE "|      |      ",endl
                    BYTE "|      O      ",endl
                    BYTE "|     /|\     ",endl
                    BYTE "|             ",endl
                    BYTE "|             ",endl
                    BYTE "+-------------+ ",endl
                    BYTE "|           | ",endl
```

```
                                    BYTE "+------------+ ",endl

HANGMAN_LIVES_01 LABEL BYTE

                                    BYTE "+------+      ",endl
                                    BYTE "|      |      ",endl
                                    BYTE "|      O      ",endl
                                    BYTE "|     /|\     ",endl
                                    BYTE "|     /       ",endl
                                    BYTE "|             ",endl
                                    BYTE "+------------+ ",endl
                                    BYTE "|           | ",endl
                                    BYTE "+------------+ ",endl

HANGMAN_LIVES_00 LABEL BYTE

                                    BYTE "+------+      ",endl
                                    BYTE "|      |      ",endl
                                    BYTE "|      O      ",endl
                                    BYTE "|     /|\     ",endl
                                    BYTE "|     / \     ",endl
                                    BYTE "|             ",endl
                                    BYTE "+------------+ ",endl
                                    BYTE "|           | ",endl
                                    BYTE "+------------+ ",endl

; random number what we generete
ranNum DWORD ?

;All words what is posible to guess.
```

```
;Pick by random generartor and put in selectedWords


manyWords     BYTE "BICYCLE",          0
                              BYTE "CANOE", 0
                              BYTE "SCATEBOARD", 0
                              BYTE "OFFSIDE", 0
                              BYTE "TENNIS", 0
                              BYTE "SOFTBALL", 0
                              BYTE "KNOCKOUT", 0
                              BYTE "CHALLENGE", 0
                              BYTE "SLALOM", 0
                              BYTE "MARATHON", 0
                              BYTE 0                                    ; End of list
len equ $ - manyWords

; number what we make to know where are you in game
statusGameLive DWORD ?

;Wordls what we select by random code
selectedWords BYTE "                ", 0
;Use as variable in funcstion for length of Array
lengthArray DWORD ?

;Letter what we guess, input from keyboard
guessLetter BYTE ?
;World what we print with -------,0
guessWords BYTE 50 DUP (?)
```

```
;Array of guess Letter
guessLetterArray BYTE 50 DUP (?)
chardelete   BYTE 'A'
;Letter what are unknows, change with -
letterDash BYTE '-'

drowDelay = 1000       ; delay 1 sec
var_loop BYTE 15       ; repeat 15 times

.code

main PROC


  ; Get the console output handle:
        INVOKE GetStdHandle, STD_OUTPUT_HANDLE
        mov consoleHandle,eax

jump_game_start_again:

  ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR message,                ; string pointer
                messageSize,                        ; string length
                ADDR bytesWritten,                  ; returns num bytes written
                0                                               ; not used
```

```
;Part of code for generate random number from 0 until 9
        mov  eax,10                 ;get random 0 to 9
        call Randomize              ;re-seed generator
        call RandomRange
        mov  ranNum,eax             ;save random number

        ;call WriteDec
        call Crlf                   ;new line


;Find a selectedWords base on generate ranNum from manyWords
        mov edx, ranNum     ;Index
        call find_str       ;Returns EDI = pointer to string, we pick world


;Copy find world in variable selectedWords
        INVOKE Str_copy,
    ADDR [edi],
    ADDR selectedWords

;Print selectedWords on screen
        ;mov edx, offset selectedWords
        ;call WriteString
        ;call Crlf                  ;new line

;Make array of dash. It would be world what we guess
        call make_array_dash
```

```
  ;Inicialization number of life what you have
        mov statusGameLive, 6

again_input_world:

  ;Print figure depending on the number of lives
        call print_hangman_live

  ;Check if you have more live. If player lost all lives, game is over
        cmp statusGameLive, 0
        je loop_game_over



        mov  eax,green+(black*16)
   call SetTextColor

        mWrite <"Guess a letter: ">

        call readChar   ;User inputs char
        cmp al, 27              ;Check if is press ESC
        je exit_main    ;YES, end game
        cmp al, 32              ;Check if is press SPACE
        je restart_game         ;YES, restart game
        and al, 0DFH    ;Convert lowercase input to uppercase.
                                        ;If uppercase, it remains uppercase
        push eax
        sub al, 'A'             ;checks if it is a letter
   cmp al, 'Z'-'A'
```

```
   jbe uppercase
        jmp again_input_world
uppercase:
        pop eax
        mov guessLetter, al
        call WriteChar
        call Crlf                ;new line
        call Crlf                ;new line

        mov  eax,white+(black*16)
   call SetTextColor



        ;Check if letter is alredy guessed
        mov ecx, LENGTHOF guessLetterArray
        mov edi, offset guessLetterArray
        mov al, guessLetter          ; Load character to find
        repne scasb                  ; Search
        je loop_guess_letter_exists                  ; Letter already exist


        call make_array_guess_letter


        ;Check if letter is in selectedWords. If not take life
        mov ecx, LENGTHOF selectedWords
        mov edi, offset selectedWords
        mov al, guessLetter          ; Load character to find
```

```
        repne scasb                     ; Search
        jne loop_take_live                                      ; Letter exist take life


  ; We are making new array, guess letter whange dash on right pleace
    mov esi, offset selectedWords       ; Source
    mov edi, offset guessWords          ; Destination
    mov ecx, LENGTHOF selectedWords     ; Number of bytes to check
    mov al, guessLetter                 ; Search for that character
    xor ebx, ebx                    ; Index EBX = 0

ride_hard_loop:
    cmp [esi+ebx], al               ; Compare memory/register
    jne @F                          ; Skip next line if no match
    mov [edi+ebx], al               ; Hang 'em lower
    @@:
    inc ebx                         ; Increment pointer
    dec ecx                         ; Decrement counter
    jne ride_hard_loop              ; Jump if ECX != 0


  ;Is there more letter to guess of we finish
        mov ecx, LENGTHOF guessWords
    mov edi, offset guessWords
    mov al, letterDash              ; Load character to find
    repne scasb                                             ; Search
    jne loop_game_win                                   ; No more letter
        jmp again_input_world                                   ; Guess next world
```

```
exit_main:

        INVOKE ExitProcess,0

mWrite <"Guess a letter: ">

        call readChar   ;User inputs char
        cmp al, 27              ;Check if is press ESC
        je exit_main    ;YES, end game
        cmp al, 32              ;Check if is press SPACE
        je restart_game         ;YES, restart game
        and al, 0DFH    ;Convert lowercase input to uppercase.
                                        ;If uppercase, it remains uppercase
        push eax
        sub al, 'A'             ;checks if it is a letter
   cmp al, 'Z'-'A'
   jbe uppercase
        jmp again_input_world
uppercase:
        pop eax
        mov guessLetter, al
        call WriteChar
        call Crlf               ;new line
        call Crlf               ;new line

        mov  eax,white+(black*16)
```

```
call SetTextColor


        ;Check if letter is alredy guessed
        mov ecx, LENGTHOF guessLetterArray
        mov edi, offset guessLetterArray
        mov al, guessLetter             ; Load character to find
        repne scasb                     ; Search
        je loop_guess_letter_exists                     ; Letter already exist


        call make_array_guess_letter


        ;Check if letter is in selectedWords. If not take life
        mov ecx, LENGTHOF selectedWords
        mov edi, offset selectedWords
        mov al, guessLetter             ; Load character to find
        repne scasb                     ; Search
        jne loop_take_live                                      ; Letter exist take life


; We are making new array, guess letter whange dash on right pleace
  mov esi, offset selectedWords       ; Source
  mov edi, offset guessWords          ; Destination
  mov ecx, LENGTHOF selectedWords     ; Number of bytes to check
  mov al, guessLetter               ; Search for that character
  xor ebx, ebx                    ; Index EBX = 0
```

```
ride_hard_loop:
   cmp [esi+ebx], al              ; Compare memory/register
   jne @F                         ; Skip next line if no match
   mov [edi+ebx], al              ; Hang 'em lower
   @@:
   inc ebx                        ; Increment pointer
   dec ecx                        ; Decrement counter
   jne ride_hard_loop             ; Jump if ECX != 0


  ;Is there more letter to guess of we finish
          mov ecx, LENGTHOF guessWords
   mov edi, offset guessWords
   mov al, letterDash             ; Load character to find
   repne scasb                                                  ; Search
   jne loop_game_win                                  ; No more letter
          jmp again_input_world                                 ; Guess next world


exit_main:

          INVOKE ExitProcess,0


                mov  edx, OFFSET guessLetterArray
                call StrLength
                mov  lengthArray, eax
```

```
                mov edi, offset guessLetterArray ; Destination
                add edi, lengthArray
                dec edi
                ;INVOKE Str_trim, ADDR guessLetterArray, guessLetter
                INVOKE Str_trim, ADDR guessLetterArray, [edi]

                cmp edi, offset guessLetterArray
                jne restart_game

 ;Return white color again
                mov  eax,white+(black*16)
                call SetTextColor
                call Crlf                       ;new line

                jmp jump_game_start_again               ; Guess next letter

loop_game_win:

        mGotoxy 0, 15

 ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_GOODGAME_00,   ; string pointer
                messageSizeGoodGame,            ; string length
                ADDR bytesWritten,              ; returns num bytes written
                0                                       ; not used
```

```
        mov eax, drowDelay
        call Delay
        mGotoxy 0, 15
        mov  eax,green+(black*16)
  call SetTextColor

; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_GOODGAME_01,   ; string pointer
                messageSizeGoodGame,            ; string length
                ADDR bytesWritten,                  ; returns num bytes written
                0                                                   ; not used

        mov eax, drowDelay
        call Delay
        mGotoxy 0, 15
        mov  eax,yellow+(black*16)
  call SetTextColor

; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_GOODGAME_02,   ; string pointer
                messageSizeGoodGame,            ; string length
                ADDR bytesWritten,                  ; returns num bytes written
                0                                                   ; not used
```

```
        mov eax, drowDelay
        call Delay
        mGotoxy 0, 15
        mov  eax,cyan+(black*16)
  call SetTextColor

; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_GOODGAME_03,   ; string pointer
                messageSizeGoodGame,                ; string length
                ADDR bytesWritten,                  ; returns num bytes written
                0                                                   ; not used

        mov eax, drowDelay
        call Delay
        mGotoxy 0, 15
        mov  eax,red+(black*16)
  call SetTextColor

        dec var_loop
        cmp var_loop, 0
        jne loop_game_win

;restar game after 4*15sekunds
        jmp jump_game_start_again
```

```
loop_game_over:

        mGotoxy 0, 15

 ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_GAMEOVER_00,   ; string pointer
                messageSizeGoodGame,        ; string length
                ADDR bytesWritten,          ; returns num bytes written
                0                                       ; not used

        mov eax, drowDelay
        call Delay
        mGotoxy 0, 15
        mov  eax,green+(black*16)
  call SetTextColor

 ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_GAMEOVER_01,   ; string pointer
                messageSizeGoodGame,        ; string length
                ADDR bytesWritten,          ; returns num bytes written
                0                                       ; not used

        mov eax, drowDelay
```

```
        call Delay
        mGotoxy 0, 15
        mov  eax,yellow+(black*16)
  call SetTextColor

; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                  ;console output handle
                ADDR HANGMAN_GAMEOVER_02,   ; string pointer
                messageSizeGoodGame,        ; string length
                ADDR bytesWritten,          ; returns num bytes written
                0                                       ; not used

        mov eax, drowDelay
        call Delay
        mGotoxy 0, 15
        mov  eax,cyan+(black*16)
  call SetTextColor

; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                  ;console output handle
                ADDR HANGMAN_GAMEOVER_03,   ; string pointer
                messageSizeGoodGame,        ; string length
                ADDR bytesWritten,          ; returns num bytes written
                0                                       ; not used

        mov eax, drowDelay
```

```
        call Delay
        mGotoxy 0, 15
        mov  eax,red+(black*16)
   call SetTextColor

        dec var_loop
        cmp var_loop, 0
        jne loop_game_over

  ;restar game after 4*15sekunds
        mov  eax,white+(black*16)
   call SetTextColor
        jmp jump_game_start_again


main ENDP

find_str PROC                                   ; ARG: EDX = index
   lea edi, manyWords         ; Address of string list

   mov ecx, len               ; Maximal number of bytes to scan
   xor al, al                 ; Scan for 0

   @@:
   sub edx, 1
   jc done                    ; No index left to scan = string found
   repne scasb                ; Scan for AL
   jmp @B                     ; Next string
```

```
 done:
        ret

find_str ENDP               ; RESULT: EDI pointer to string[edx]

make_array_dash PROC
        mov  edx,OFFSET selectedWords
   call StrLength           ; Length of a null-terminated string pointed to by EDX
   mov  lengthArray,eax

   mov al, '-'              ; Default charcter for guessWords
   mov ecx, lengthArray                 ; REP counter
   mov edi, offset guessWords  ; Destination
   rep stosb                ; Build guessWords
   mov BYTE PTR [edi], 0      ; Store the null termination

   ret
make_array_dash ENDP

make_array_guess_letter PROC
        mov  edx, OFFSET guessLetterArray
   call StrLength                               ; Length of a null-terminated string pointed to by
EDX
   mov  lengthArray, eax

   mov edi, offset guessLetterArray ; Destination
   add edi, lengthArray
        mov al, guessLetter
```

```
        mov BYTE PTR [edi], al      ; Store guessLetter
        inc edi
        mov BYTE PTR [edi], ','     ; Store the null termination

    ret
make_array_guess_letter ENDP

print_hangman_live PROC

        mov eax, statusGameLive

        cmp eax, 6
        je live_6
        cmp eax, 5
        je live_5
        cmp eax, 4
        je live_4
        cmp eax, 3
        je live_3
        cmp eax, 2
        je live_2
        cmp eax, 1
        je live_1
        cmp eax, 0
        je live_0

live_6:   ; Write a string to the console:
        INVOKE WriteConsole,
```

```
        consoleHandle,                  ; console output handle
        ADDR HANGMAN_LIVES_06,     ; string pointer
        messageSizeGoodGame,        ; string length
        ADDR bytesWritten,          ; returns num bytes written
        0                                   ; not used
    call Crlf                               ; new line
    call Crlf                               ; new line
    mov edx, offset guessWords
    call WriteString        ; write a string pointed to by EDX
    call Crlf           ; new line
    call Crlf                               ; new line
    mWrite <"Guessed letter are: ">
    mov edx, offset guessLetterArray
    call WriteString        ; write a string pointed to by EDX
    call Crlf           ; new line
    call Crlf           ; new line
    ret

live_5:   ; Write a string to the console:
    INVOKE WriteConsole,
        consoleHandle,                  ;console output handle
        ADDR HANGMAN_LIVES_05,     ; string pointer
        messageSizeGoodGame,        ; string length
        ADDR bytesWritten,          ; returns num bytes written
        0                                   ; not used
    call Crlf                               ; new line
    call Crlf                               ; new line
    mov edx, offset guessWords
```

```
        call WriteString            ; write a string pointed to by EDX
        call Crlf                   ; new line
        call Crlf                                               ; new line
        mWrite <"Guessed letter are: ">
        mov edx, offset guessLetterArray
        call WriteString            ; write a string pointed to by EDX
        call Crlf                   ; new line
        call Crlf                   ; new line
        ret

live_4:  ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_LIVES_04,      ; string pointer
                messageSizeGoodGame,                ; string length
                ADDR bytesWritten,                  ; returns num bytes written
                0                                               ; not used
        call Crlf                                               ; new line
        call Crlf                                               ; new line
        mov edx, offset guessWords
        call WriteString            ; write a string pointed to by EDX
        call Crlf                   ; new line
        call Crlf                                               ; new line
        mWrite <"Guessed letter are: ">
        mov edx, offset guessLetterArray
        call WriteString            ; write a string pointed to by EDX
        call Crlf                   ; new line
        call Crlf                   ; new line
```

```
        ret

live_3:  ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_LIVES_03,     ; string pointer
                messageSizeGoodGame,            ; string length
                ADDR bytesWritten,              ; returns num bytes written
                0                                       ; not used
        call Crlf                                       ; new line
        call Crlf                                       ; new line
        mov edx, offset guessWords
        call WriteString            ; write a string pointed to by EDX
        call Crlf               ; new line
        call Crlf                                       ; new line
        mWrite <"Guessed letter are: ">
        mov edx, offset guessLetterArray
        call WriteString            ; write a string pointed to by EDX
        call Crlf               ; new line
        call Crlf               ; new line
        ret

live_2:  ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                          ;console output handle
                ADDR HANGMAN_LIVES_02,     ; string pointer
                messageSizeGoodGame,            ; string length
                ADDR bytesWritten,              ; returns num bytes written
```

```
        0                                                   ; not used
    call Crlf                                               ; new line
    call Crlf                                               ; new line
    mov edx, offset guessWords
    call WriteString        ; write a string pointed to by EDX
    call Crlf               ; new line
    call Crlf                                               ; new line
    mWrite <"Guessed letter are: ">
    mov edx, offset guessLetterArray
    call WriteString        ; write a string pointed to by EDX
    call Crlf               ; new line
    call Crlf               ; new line
    ret

live_1:  ; Write a string to the console:
    INVOKE WriteConsole,
        consoleHandle,                          ;console output handle
        ADDR HANGMAN_LIVES_01,    ; string pointer
        messageSizeGoodGame,            ; string length
        ADDR bytesWritten,              ; returns num bytes written
        0                                                   ; not used
    call Crlf                                               ; new line
    call Crlf                                               ; new line
    mov edx, offset guessWords
    call WriteString        ; write a string pointed to by EDX
    call Crlf               ; new line
    call Crlf                                               ; new line
    mWrite <"Guessed letter are: ">
```

```
        mov edx, offset guessLetterArray
        call WriteString        ; write a string pointed to by EDX
        call Crlf               ; new line
        call Crlf               ; new line
        ret

live_0:  ; Write a string to the console:
        INVOKE WriteConsole,
                consoleHandle,                  ;console output handle
                ADDR HANGMAN_LIVES_00,  ; string pointer
                messageSizeGoodGame,            ; string length
                ADDR bytesWritten,              ; returns num bytes written
                0                                       ; not used
        call Crlf                               ; new line
        call Crlf                               ; new line
        mov edx, offset guessWords
        call WriteString        ; write a string pointed to by EDX
        call Crlf               ; new line
        call Crlf                               ; new line
        mWrite <"Guessed letter are: ">
        mov edx, offset guessLetterArray
        call WriteString        ; write a string pointed to by EDX
        call Crlf               ; new line
        call Crlf               ; new line
        ret

print_hangman_live ENDP
```

END main