1. Create a procedure named *Scan_String* to find the index of the first occurrence of the character '#' in the given string.

Str1 BYTE '127&j~3#^&*#*#45^',0

```
TITLE Scan string
INCLUDE Irvine32.inc
.data
Str1 BYTE "127&j~3#^&*#*#45^",0
msgfound BYTE "Index of character found is: ", 0
msgnotfound BYTE "Not found",0
.code
main PROC
mov edi,OFFSET Str1 ; EDI points to the string
mov al,'#'          ; search for #
call Scan_string
exit
main ENDP
```

```
Scan_string PROC
mov ecx,LENGTHOF Str1 ; set the search count
Cld                   ; direction = forward
repne scasb           ; repeat while not equal
jnz quit              ; quit if letter not found
dec edi               ; found: back up EDI
mov edx, OFFSET msgfound
call Writestring
mov eax, 0
sub edi, OFFSET Str1
mov eax, edi
call Writedec
jmp _exit
quit:
mov edx, OFFSET msgnotfound
call WriteString
_exit:
ret
Scan_string ENDP
END main
```

```
Index of character found is: 7
```

2. Modify the above procedure to take *offset of string1* and the *character to be searched* as argument.

Arguments pushed onto stack and accessed through explicit stack parameter method.

```
TITLE Scan string
INCLUDE Irvine32.inc
.data
Str1 BYTE "127&j~3#^&*#*#45^",0
msgfound BYTE "Index of character found is: ", 0
msgnotfound BYTE "Not found",0
.code
main PROC
push OFFSET Str1 ; EDI points to the string
push '#'          ; search for #
call Scan_string
call Crlf
exit
main ENDP
```

```
Scan_string PROC
push ebp
mov ebp, esp
mov edi, [ebp+12]
mov al, [ebp+8]
mov ecx, LENGTHOF Str1
Cld                 ; direction = forward
repne scasb         ; repeat while not equal
jnz quit            ; quit if letter not found
dec edi             ; found: back up EDI
mov edx, OFFSET msgfound
call Writestring
mov eax, 0
sub edi,[ebp+12]
mov eax, edi
call Writedec
jmp _exit
```

```
quit:
mov edx, OFFSET msgnotfound
call WriteString
_exit:
pop ebp
ret 8
Scan_string ENDP
END main
```

```
Index of character found is: 7
```

3. Create *IsCompare* procedure to compare two strings.

```
TITLE Q3
INCLUDE Irvine32.inc
.data
source BYTE "MARTIN ",0
dest BYTE "MARTINEZ",0
str1 BYTE "Source is smaller",0
str2 BYTE "Source is not smaller",0
.code
main PROC
cld ; direction = forward
mov esi,OFFSET source
mov edi,OFFSET dest
mov ecx,LENGTHOF source
push esi
push edi
push ecx
call IsCompare
exit
main ENDP
```

```
IsCompare PROC
push ebp
mov ebp, esp
mov esi, [ebp+16]
mov edi, [ebp+12]
mov ecx, [ebp+8]
repe cmpsb
jb source_smaller
mov edx,OFFSET str2 ; "source is not smaller"
jmp done
source_smaller:
mov edx,OFFSET str1 ; "source is smaller"
done:
call WriteString
pop ebp
ret 12
IsCompare ENDP

END main
```

```
Source is smaller
```

4. Create a Str_Reverse procedure to reverse strings.

```
TITLE Q4
INCLUDE Irvine32.inc
.data
inputStr BYTE "HELLO", 0
msg BYTE "Reversed string: ", 0
.code
main PROC
mov edx, OFFSET inputStr
call WriteString
call CrLf
push OFFSET inputStr
call Str_Reverse

mov edx, OFFSET msg
call WriteString
mov edx, OFFSET inputStr
call WriteString
call CrLf
exit
main ENDP
```

```
Str_Reverse PROC
    push ebp
    mov ebp, esp

    mov esi, [ebp+8]
    mov edi, esi

find_end:
    cmp BYTE PTR [edi], 0
    je reverse_start
    inc edi
    jmp find_end
```

```
reverse_start:
    dec edi             ; Move back to the last character
reverse_loop:
    cmp esi, edi        ; Check if pointers have crossed
    jge reverse_done    ; If yes, reversing is complete

    mov al, [esi]
    mov bl, [edi]
    mov [esi], bl
    mov [edi], al

    inc esi
    dec edi
    jmp reverse_loop

reverse_done:
    pop ebp
    ret 4
Str_Reverse ENDP

END main
```

```
HELLO
Reversed string: OLLEH
```

5. Create a procedure that Loads an array of integer by multiplying it with Load(offset array, byte no)

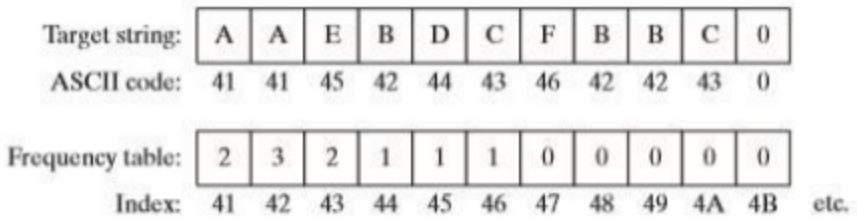
```
INCLUDE Irvine32.inc

.data
array DWORD 5, 10, 15, 20, 25
msg BYTE "new array: ", 0

.code

Load PROC arr:DWORD, num:BYTE
    mov esi, arr
    movzx ebx, num
l1:
    mov eax, [esi]
    imul eax, ebx
    mov [esi], eax
    add esi, 4
    loop l1
    ret
Load ENDP
```

```
main PROC
    mov ecx, LENGTHOF array
    INVOKE Load, ADDR array, 2

    mov edx, OFFSET msg
    call WriteString

    mov esi, OFFSET array
    mov ecx, LENGTHOF array

print:
    mov eax, [esi]
    call WriteDec
    mov al,' '
    call writechar
    add esi, 4
    loop print

    exit
main ENDP
```

```
new array: 10 20 30 40 50
C:\Users\k230842\source\repos\COALl
```

6. Write the procedure to get_frequency Find the frequency of characters:

```
.data
        target BYTE "AAEBDCFBBC",0
        freqTable DWORD 256 DUP(0)
.code
        INVOKE Get_frequencies, ADDR target, ADDR freqTable
```



Target string:	A	A	E	B	D	C	F	B	B	C	0	
ASCII code:	41	41	45	42	44	43	46	42	42	43	0	
Frequency table:	2	3	2	1	1	1	0	0	0	0	0	
Index:	41	42	43	44	45	46	47	48	49	4A	4B	etc.

```
INCLUDE Irvine32.inc

.data
target BYTE "AAEBDCFBBC", 0
freqTable DWORD 256 DUP(0)

.code
Get_frequencies PROC targ:DWORD, freq :DWORD
    cld
    mov esi, targ
    mov edi, freq

    mov ecx,0

count:
    mov al, [esi + ecx]
    test al, al
    jz done

    movzx eax, al       ;ascii val
    inc dword ptr [edi + eax*4]

    inc ecx
    jmp count
```

```
done:
    ret
Get_frequencies ENDP
main PROC
    INVOKE Get_frequencies, ADDR target, ADDR freqTable
    mov edx,offset target
    call writestring
    call crlf
    mov edi ,offset target
    mov ecx, 0
    print:
        mov eax, [freqTable + ecx*4]
        cmp eax, 0
        je skip
        mov eax, [freqTable + ecx*4]
        call WriteDec
        mov al,' '
        call writechar
    skip:
        inc ecx
        inc edi
        cmp ecx, 256
        jl print
    exit
main ENDP
END main
```

No issues found

```
AAEBDCFBBC
2 3 2 1 1 1
C:\Users\k230842\source\repos\COALlab10\
To automatically close the console when
le when debugging stops.
Press any key to close this window . . .
```