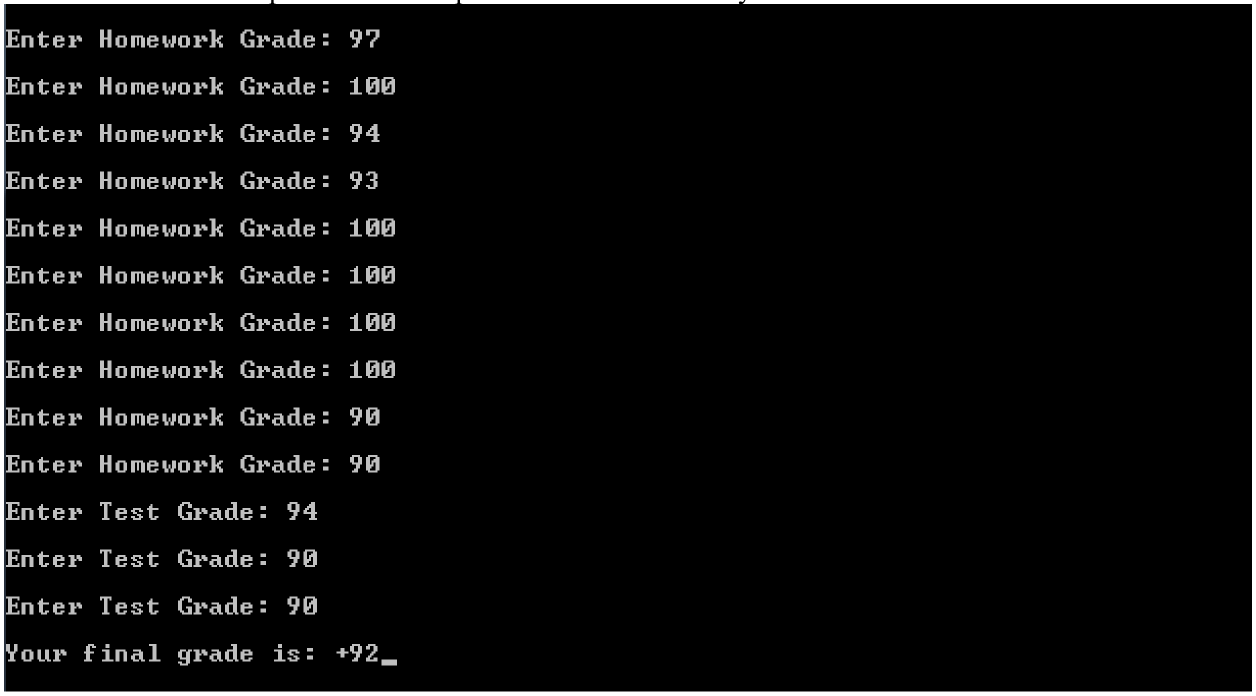
HW 10: Strings/Fun Programming

1. Draft a program that will ultimately calculate your current grade in this class. You may choose to use dummy values for the input. The program/procedure specifics are left to you. However, your program must do the following:
   1) Take user input for homework grades (10 total),
   2) Take user input for exam grades (2 or 3),
   3) Calculate the average for homework grades and stores the value in a variable named *HWAverage*,
   4) Calculate the average of *HWAverage* and the test scores (as these are weighted equally), and;
   5) Print the weighted average to the screen. As the result will likely not be an integer, feel free to round up. I leave the exploration of that idea to you.



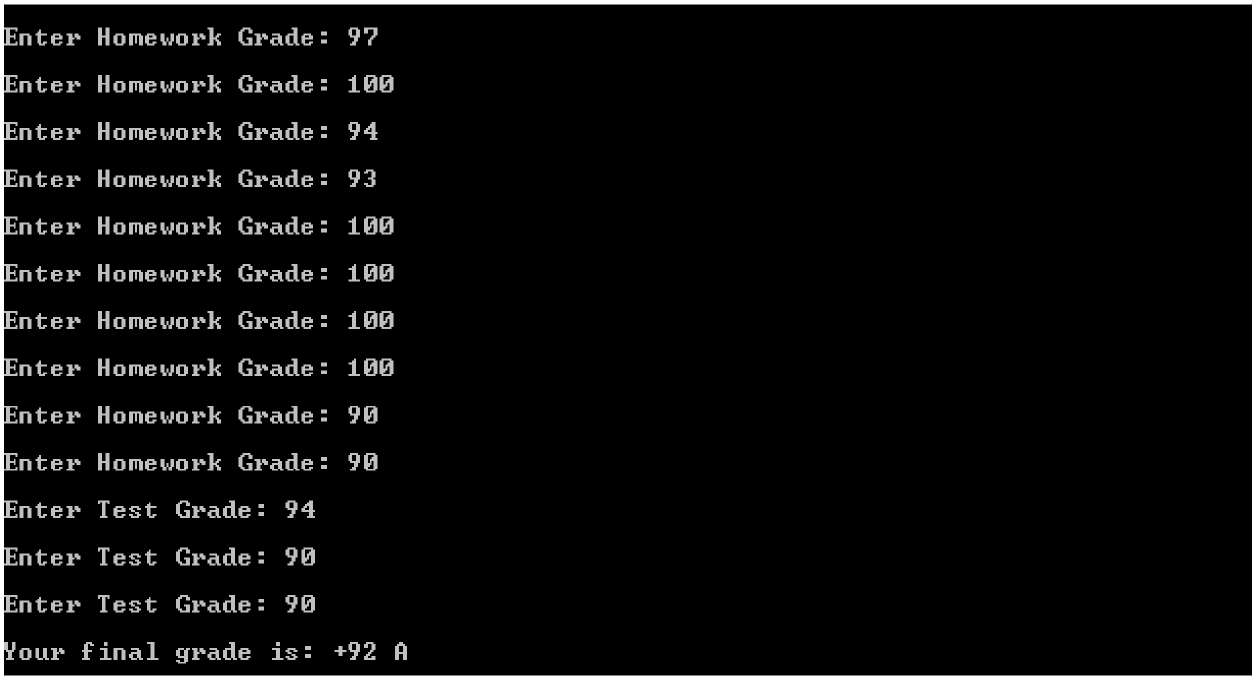
```
Enter Homework Grade: 97
Enter Homework Grade: 100
Enter Homework Grade: 94
Enter Homework Grade: 93
Enter Homework Grade: 100
Enter Homework Grade: 100
Enter Homework Grade: 100
Enter Homework Grade: 100
Enter Homework Grade: 90
Enter Homework Grade: 90
Enter Test Grade: 94
Enter Test Grade: 90
Enter Test Grade: 90
Your final grade is: +92_
```

```
INCLUDE Irvine32.inc

.data
homework WORD 0
HWAverage WORD 0
tests WORD 0
TestAverage WORD 0
TotalGrade WORD 0
hwPrompt BYTE 0Dh, 0Ah, "Enter Homework Grade: ", 0h
testPrompt BYTE 0Dh, 0Ah, "Enter Test Grade: ", 0h
result BYTE 0Dh, 0Ah, "Your final grade is: ", 0h

.code
main PROC
        PUSH 10                         ;set number of HW grades to be input

        Call HWCalc                     ;call HW average calculator

        PUSH 3                          ;set number of test grades to be input

        Call TestCalc                   ;call test average calculator

        PUSH 25                         ;set HW weight
        PUSH 75                         ;set test weight
        Call GetGrade                       ;call proc that calculates total grade
exit
main ENDP

HWCalc PROC
        PUSH ebp                            ;save ebp
        MOV ebp, esp                        ;copy stack pointer to ebp

        MOV ecx, DWORD PTR [ebp + 8]        ;look back 8 bytes

        MOV edx, OFFSET hwPrompt            ;set prompt to be output

        L1:
                Call WriteString            ;prompt
                Call ReadInt                ;read in grade

                ADD homework, ax            ;add new HW grade to aggregator
        LOOP L1

        MOV dx, 0                           ;clear dx
        MOV ax, homework                    ;copy HW total to ax
        MOV bx, 0Ah                         ;set divisor
        DIV bx                              ;divide total by 10

        MOV HWAverage, ax                   ;copy result into HWAverage

        cmp dx, 4                           ;check for remainder >= 5
        JG ROUND                            ;round up if ^^
        POP ebp
        RET 4

        ROUND:
        INC HWAverage
        POP ebp
        RET 4
HWCalc ENDP

TestCalc PROC
        PUSH ebp                            ;save ebp
        MOV ebp, esp                        ;copy stack pointer to ebp

        MOV ecx, DWORD PTR [ebp + 8]        ;look back 8 bytes

        MOV edx, OFFSET testPrompt          ;set prompt to be output

        L1:
                Call WriteString            ;prompt
                Call ReadInt                ;read in grade

                ADD tests, ax               ;add new test grade to aggregator
        LOOP L1

        MOV dx, 0                           ;clear dx
        MOV ax, tests                       ;copy test total to ax
        MOV bx, 3                           ;set divisor
        DIV bx                              ;divide total by 3

        MOV TestAverage, ax                 ;copy result into TestAverage

        CMP dx, 1                           ;check for remainder >= 2
        JG ROUND                            ;round up if ^^
        POP ebp
        RET 4

        ROUND:
        INC TestAverage
        POP ebp
        RET 4
TestCalc ENDP
```

```
GetGrade PROC
        PUSH ebp                            ;save ebp
        MOV ebp, esp                        ;copy stack pointer to ebp

        MOV ebx, DWORD PTR [ebp + 8]        ;look back 6 bytes for test weight

        MOV ax, TestAverage                 ;copy test average to ax
        MUL bx                              ;multiply test average by test weight
        MOV dx, 0                           ;clear dx
        MOV bx, 100d                        ;set divisor
        DIV bx                              ;divide new test total by 100d

        MOV TotalGrade, ax

        MOV cx, dx                          ;save remainder

        MOV ebx, DWORD PTR [ebp + 12]       ;look back 8 bytes for HW weight
        MOV ax, HWAverage                   ;copy HW average to ax
        MUL bx                              ;multiply HW average by HW weight
        MOV dx, 0                           ;clear dx
        MOV bx, 100d                        ;set divisor
        DIV bx                              ;divide new HW total by 100d

        ADD TotalGrade, ax

        ADD dx, cx
        CMP dx, 49                          ;check for remainder >= 50
        JG ROUND2                           ;round up if ^^
        JL CONTINUE                         ;else jump to CONTINUE

        ROUND2:
        INC TotalGrade

        CONTINUE:
        MOV edx, OFFSET result
        Call WriteString

        MOVZX eax, TotalGrade
        Call WriteInt

        POP ebp
        RET 8
GetGrade ENDP
END main
```

2. Add functionality to the program drafted above that will compare the calculated average in this class to the data below. The amended program should display not only your numerical grade, but also the letter grade associated. You should reference the section of the text that discusses Table Driven Selection. Use the following data as a guide for letter grade and score range association:

Score Range	Letter Grade
90 – 100	A
80 – 89	B
70 - 79	C
60 – 69	D
0 - 59	F



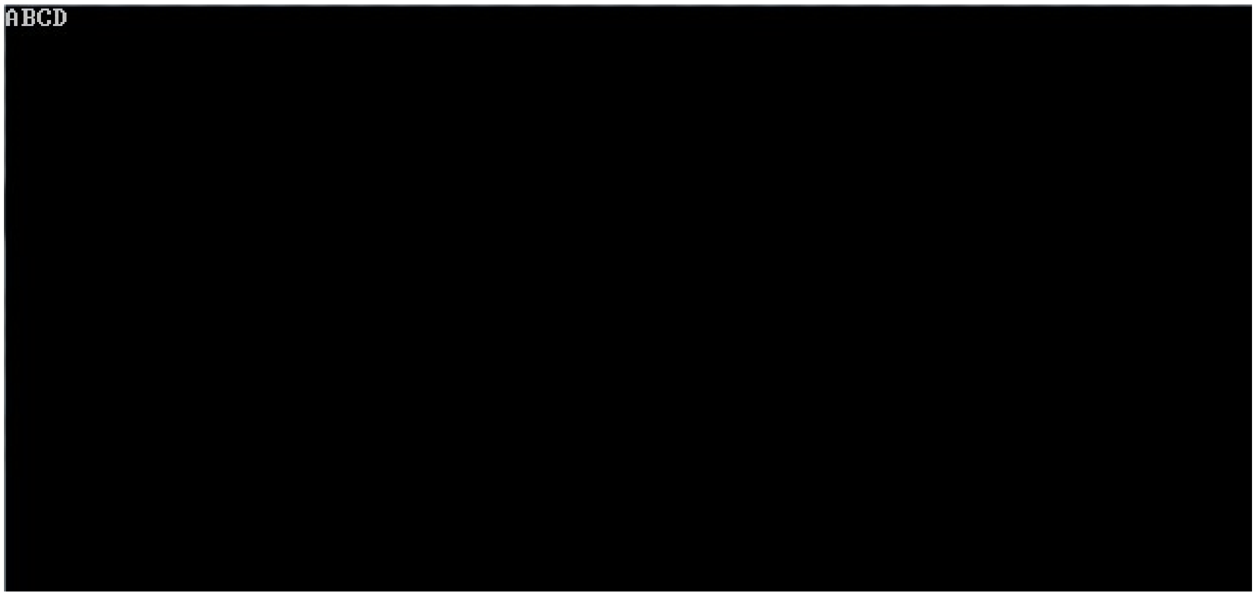
```
Enter Homework Grade: 97

Enter Homework Grade: 100

Enter Homework Grade: 94

Enter Homework Grade: 93

Enter Homework Grade: 100

Enter Homework Grade: 100

Enter Homework Grade: 100

Enter Homework Grade: 100

Enter Homework Grade: 90

Enter Homework Grade: 90

Enter Test Grade: 94

Enter Test Grade: 90

Enter Test Grade: 90

Your final grade is: +92 A
```

```
.data
homework WORD 0
HWAverage WORD 0
tests WORD 0
TestAverage WORD 0
TotalGrade WORD 0
hwPrompt BYTE 0Dh, 0Ah, "Enter Homework Grade: ", 0h
testPrompt BYTE 0Dh, 0Ah, "Enter Test Grade: ", 0h
result BYTE 0Dh, 0Ah, "Your final grade is: ", 0h

.code
main PROC
        PUSH 10                         ;set number of HW grades to be input

        Call HWCalc                     ;call HW average calculator

        PUSH 3                          ;set number of test grades to be input

        Call TestCalc                   ;call test average calculator

        PUSH 25                         ;set HW weight
        PUSH 75                         ;set test weight

        Call GetGrade                   ;call proc that calculates total grade
exit
main ENDP

HWCalc PROC
        PUSH ebp                        ;save ebp
        MOV ebp, esp                    ;copy stack pointer to ebp

        MOV ecx, DWORD PTR [ebp + 8]    ;look back 8 bytes

        MOV edx, OFFSET hwPrompt        ;set prompt to be output

        L1:
                Call WriteString        ;prompt
                Call ReadInt            ;read in grade

                ADD homework, ax        ;add new HW grade to aggregator
```

```
          LOOP L1

          MOV dx, 0                             ;clear dx
          MOV ax, homework                      ;copy HW total to ax
          MOV bx, 0Ah                           ;set divisor
          DIV bx                                ;divide total by 10

          MOV HWAverage, ax                     ;copy result into HWAverage

          cmp dx, 4                             ;check for remainder >= 5
          JG ROUND                              ;round up if ^^
          POP ebp
          RET 4

          ROUND:
          INC HWAverage
          POP ebp
          RET 4
HWCalc ENDP

TestCalc PROC
          PUSH ebp                              ;save ebp
          MOV ebp, esp                          ;copy stack pointer to ebp

          MOV ecx, DWORD PTR [ebp + 8]          ;look back 8 bytes

          MOV edx, OFFSET testPrompt            ;set prompt to be output

          L1:
                    Call WriteString            ;prompt
                    Call ReadInt                ;read in grade

                    ADD tests, ax               ;add new test grade to aggregator
          LOOP L1

          MOV dx, 0                             ;clear dx
          MOV ax, tests                         ;copy test total to ax
          MOV bx, 3                             ;set divisor
          DIV bx                                ;divide total by 3

          MOV TestAverage, ax                   ;copy result into TestAverage

          CMP dx, 1                             ;check for remainder >= 2
          JG ROUND                              ;round up if ^^
          POP ebp
          RET 4

          ROUND:
          INC TestAverage
          POP ebp
          RET 4
TestCalc ENDP

GetGrade PROC
          PUSH ebp                              ;save ebp
          MOV ebp, esp                          ;copy stack pointer to ebp

          MOV ebx, DWORD PTR [ebp + 8]          ;look back 6 bytes for test weight

          MOV ax, TestAverage                   ;copy test average to ax
          MUL bx                                ;multiply test average by test weight
          MOV dx, 0                             ;clear dx
          MOV bx, 100d                          ;set divisor
          DIV bx                                ;divide new test total by 100d

          MOV TotalGrade, ax

          MOV cx, dx                            ;save remainder

          MOV ebx, DWORD PTR [ebp + 12]         ;look back 8 bytes for HW weight
```

```
		MOV ax, HWAverage			;copy HW average to ax
		MUL bx					;multiply HW average by HW weight
		MOV dx, 0				;clear dx
		MOV bx, 100d				;set divisor
		DIV bx					;divide new HW total by 100d

		ADD TotalGrade, ax

		ADD dx, cx
		CMP dx, 49				;check for remainder >= 50
		JG ROUND2				;round up if ^^
		JL CONTINUE				;else jump to CONTINUE

		ROUND2:
		INC TotalGrade

		CONTINUE:
		MOV edx, OFFSET result
		Call WriteString

		MOVZX eax, TotalGrade
		Call WriteInt

		POP ebp
		RET 8
		Call LetterGrade			;calls procedure that prints appropriate letter grade
		RET 8
GetGrade ENDP

LetterGrade PROC
		MOV al, ' '
		Call WriteChar

		MOVZX eax, TotalGrade
		cmp eax, 60
		jl FGRADE
		cmp eax, 70
		jl DGRADE
		cmp eax, 80
		jl CGRADE
		cmp eax, 90
		jl BGRADE

		MOV al, 'A'
		Call WriteChar
		RET

		BGRADE:
		MOV al, 'B'
		Call WriteChar
		RET

		CGRADE:
		MOV al, 'C'
		Call WriteChar
		RET

		DGRADE:
		MOV al, 'D'
		Call WriteChar
		RET

		FGRADE:
		MOV al, 'F'
		Call WriteChar
		RET
LetterGrade ENDP
END main
```

3. Chapter 9, section 7 contains a Str_copy procedure. Draft a modified procedure that limits the number of characters to be copied. Please embed your code in your homework submission along with a screenshot post execution.



```
ABCD
```

```
INCLUDE Irvine32.inc

Mod_copy PROTO,
        source:PTR BYTE,                ; source string
        target:PTR BYTE                 ; target string

Str_length PROTO,
        pString:PTR BYTE                ; pointer to string

.data
string_1 BYTE "ABCDEFG",0
string_2 BYTE 100 DUP(?)
limit BYTE 4

.code
main PROC
        call Clrscr

        INVOKE Mod_copy,                ; copy string_1 to string_2
          OFFSET string_1,
          OFFSET string_2

        mov  edx,OFFSET string_2
        call WriteString
        call Crlf

        exit
main ENDP

Mod_Copy PROC USES eax ecx esi edi,
        source: PTR BYTE,
        target: PTR BYTE

        ;INVOKE Str_length, source
        ;MOV ecx, eax
        ;INC ecx
        MOVZX ecx, limit                ;instead of using source length, use a limit
        MOV esi, source
        MOV edi, target
        cld
        REP MOVSB
        RET
Mod_Copy ENDP
END main
```