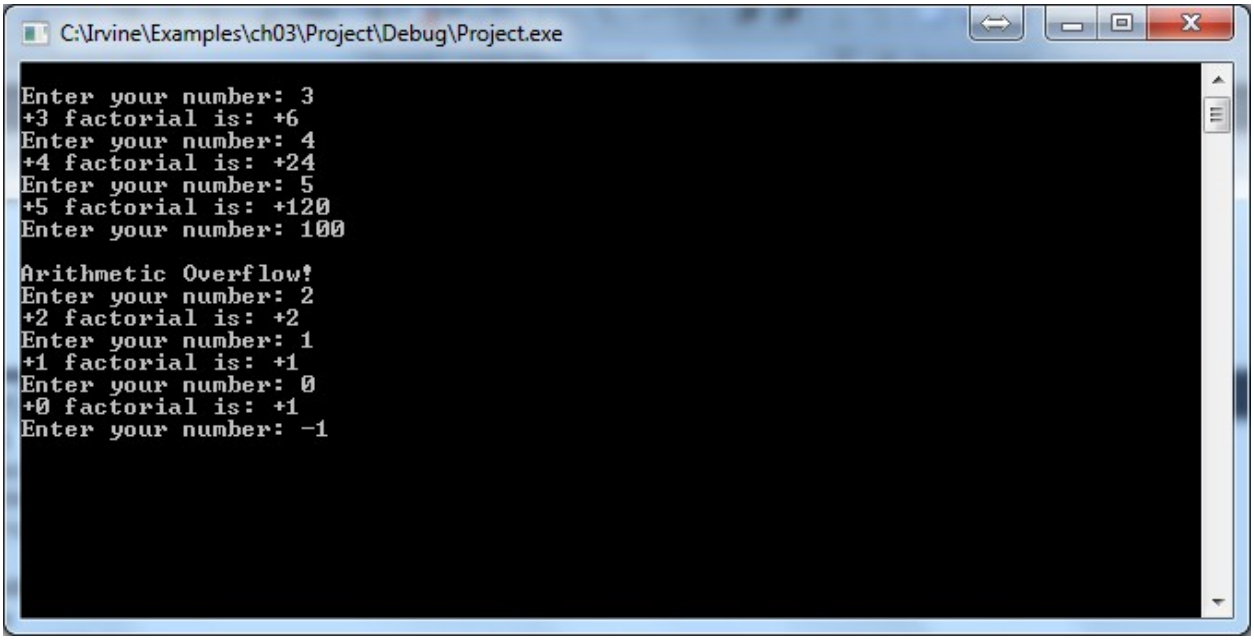
HW 9: Theme: Advanced Procedures, Stack Parameters and Frames, Local, BCD

1. Draft a program that calculates *n!*. Your program must do the following:
   1) Request user input for *n* via prompt;
   2) Calculate *n* factorial;
   3) Print the value of *n!* to the screen with a result message;
   4) Continuously ask for input, until any negative value is supplied which will be the input for exiting the program;
   5) Return an error message if an arithmetic overflow occurs resulting from evaluating *n!*; and
   6) All error messages should re-prompt the user for input.

   Please embed your code into your homework solution along with a screen shot illustrating the manner in which your program handles proper and improper user input.

```
INCLUDE Irvine32.inc
.data
        n DWORD 0
        prompt BYTE 0Dh, 0Ah, "Enter your number: ", 0
        error BYTE 0Dh, 0Ah, "Arithmetic Overflow!", 0
        result BYTE " factorial is: ", 0
.code
main PROC
        BEGIN:
        MOV edx, OFFSET prompt          ;load prompt
        call WriteString                ;write prompt
        call ReadInt                    ;read in n

        cmp eax, 0                      ;check for negative number
        jl FINISH                       ;if n is < 0, exit
        cmp eax, 2                      ;check for special number
        jl SPECIAL                      ;if n = 0,1 jump to SPECIAL

        MOV n, eax                      ;copy eax into n
        MOV ebx, eax                    ;copy same value into ebx

        MOV ecx, n                      ;set loop counter
        DEC ecx                         ;to n - 1
        L1:
                DEC ebx                 ;decrement ebx
                MUL ebx                 ;multiply eax by next integer
                jo OVER                 ;if result oversteps bounds of eax (overflows into edx), jump to OVER
                LOOP L1                 ;next iteration
        jmp SUCCESS                     ;if no error jumps have been made, jump to SUCCESS

        OVER:
        MOV edx, OFFSET error           ;load error
        call WriteString                ;write error
        jmp BEGIN                       ;jump back to beginning

        SPECIAL:
        MOV n, eax                      ;reset n
        MOV eax, 1                      ;n! is 1

        SUCCESS:
        MOV ebx, eax                    ;save numerical result
        MOV eax, n                      ;load n
        MOV edx, OFFSET result          ;load result
        call WriteInt                   ;write n
        call WriteString                ;write result
        MOV eax, ebx                    ;load numerical result
        call WriteInt                   ;write numerical result
        jmp BEGIN                       ;jump back to beginning

        FINISH:
exit
main ENDP
END main
```

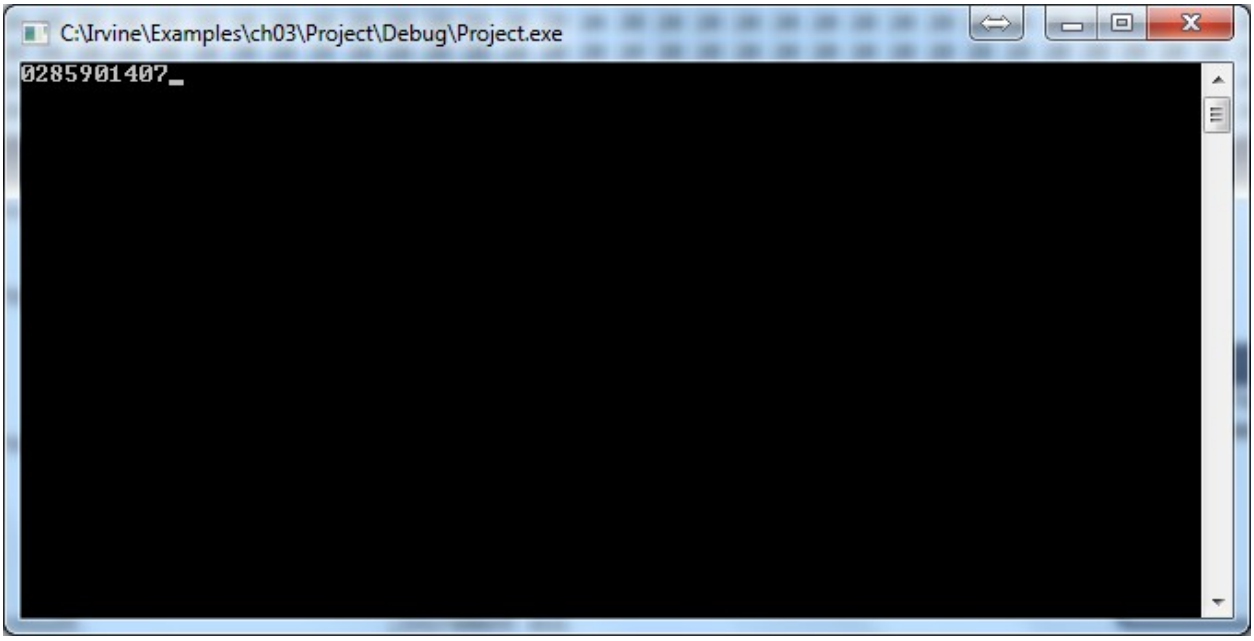


2. Draft a program that subtracts two BCD numbers (9-digits each). The first BCD number is stored in an array named *myAuburnID*, and the second in an array named *myAuburnID-Inverted*. The first number is your actual Auburn ID number; the second is the reversal of that number. Your program should do the following:
   1) Display contents of the memory before program execution,
   2) Subtract *myAuburnID-Inverted* from *myAuburnID*,
   3) Store the difference in a variable named *result*, and;
   4) Display contents of memory post execution.

   Please embed your code into your homework solution along with a screen shot post execution.

```
INCLUDE Irvine32.inc

.data
myAuburnID BYTE 09h,02h,43h,56h,16h
myAuburnIDInverted BYTE 06h,16h,53h,42h,09h
result BYTE 5 DUP (0)

.code
main PROC

        MOV ebx, LENGTHOF myAuburnId                    ;initialize offset
        MOV ecx, LENGTHOF myAuburnId                    ;initialize counter
        clc                                             ;clear carry flag

        L1:
                DEC ebx                                 ;decrement offset
                MOV al, BYTE PTR [myAuburnID + ebx]     ;store next two BCD digits in al
                SBB al, BYTE PTR [myAuburnIDInverted + ebx]  ;subtract (with borrow) from myAuburnIDInverted
                DAS                                     ;convert result from hex to BCD
                MOV BYTE PTR [result + ebx], al         ;store result
                LOOP L1

        MOV ecx, LENGTHOF result                        ;initialize ecx to length of result
        MOV esi, OFFSET result                          ;initialize esi to result

        L2:
                MOV al, BYTE PTR [esi]                  ;move first digit to al
                SHR al, 4                               ;Just first BCD digit
                OR al, 30h                              ;convert to ASCII
```

```
                Call WriteChar                  ;write ASCII char

                MOV al, BYTE PTR [esi]          ;move first digit to al
                SHL al, 4                       ;Just second BCD digit
                SHR al, 4                       ;put second BCD digit back to end of AL
                OR al, 30h                      ;convert to ASCII
                Call WriteChar                  ;write ASCII char

                ADD esi, TYPE result            ;increment esi
        LOOP L2
exit
main ENDP
END main
```



3. Draft a procedure called *SubThree* that is similar to the *AddTwo* program within the slides (at or about #30). *SubThree* receives three parameters x, y and z from the stack and outputs a value equal to x – y – z. In order to receive credit, your submission must do the following:
   1) Pass parameters x, y, and z by value on the stack using the push instruction. Use x= 207, y = 68 and z = 39,
   2) *SubThree* must get its inputs from the stack,
   3) Call *SubThree* and print the result to the screen.

   Please embed your code into your homework solution along with a screen shot post execution.

```
INCLUDE Irvine32.inc
.data
        x DWORD 207
        y DWORD 68
        z DWORD 39
.code
main PROC
        PUSH x                          ;push x
        PUSH y                          ;push y
        PUSH z                          ;push z
```

```
        call SubThree                   ;call SubThree procedure

        call WriteInt                   ;print result
exit
main ENDP

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

        MOV eax, DWORD PTR [ebp + 16]   ;look back 16 bytes (x)
        MOV ebx, DWORD PTR [ebp + 12]   ;look back 12 bytes (y)
        sub eax, ebx                    ;eax = x - y
        MOV ebx, DWORD PTR [ebp + 8]    ;look back 8 bytes (z)
        sub eax, ebx                    ;eax = x - y - z

        POP ebp                         ;restore ebp
        ret 12                          ;upon return, delete 12 bytes from stack (x,y,z)
SubThree ENDP
END main
```