### CS 278 Exam 1: Hands on Practical Problem – DUE 2/27 SCORE:\_\_\_\_\_\_/40

In your CS 278 folder on CS1, make a folder called EXAM1. Inside this folder, place an assembler project called **cards**. Starting with the code given below, follow the instructions to initialize memory as described. Most of the program has already been written for you. **This exam is, in part, a learning exercise in its own right.** The code, below, will work as-written if you make the modifications required, but, while the modifications you must make have been covered in class, not all parts of the code given you has. Your mission will be to make the modifications called for while aiming to understand, but not to change, the code I’ve supplied you. Submit to the EXAM1 directory on WhitGit.

You must provide 13 symbolic integer constants: UNO=1, DOS=2, TRES=3, QUATRO=4, CINCO=5, SEIS=6, SIETE=7, OCHO=8, NUEVE=9, DIES=10, ONCE=11, DOCE=12, TRECE=13 and then use these specific symbolic constants to initialize an area of memory called **numeros**.

Put your name and date here.

TITLE MASM Template **(**main.asm**)**

Put integer constants here before the data segment

; Description: Number Value Program

; Author:

; Date:

Use the integer constants defined above to initialize memory for cards

INCLUDE Irvine32.inc

; TODO: PLACE SYMBOLIC INTEGER CONSTANTS HERE

;

.data

; TODO: MAKE A LABEL numeros THAT LOCATES MEMORY FOR ALL THE DWORD NUMBER VALUES

; USE THE INTEGER CONSTANTS SPECIFIED ABOVE TO INITALIZE ALL 13 DWORD VALUES

; PUT THE NUMBER CONSTANTS IN MEMORY IN DESCENDING ORDER SO THAT THE OUTPUT FROM YOUR

; PROGRAM MATCHES THAT SHOWN BELOW

.code

main proc

; The code for main has already been written for you...

; Setup a loop that will run 13 times and output all the DWORD number values

**mov** **ecx,** 13

**mov** **esi,** 0

label1**:**

; This writes out each DWORD from the array "numeros" to the screen

**mov** **eax,**numeros**[esi\***TYPE **DWORD]**

**call** WriteInt

; This increments the array index counter

**inc** **esi**

**LOOP** label1 ; Loop 13 times (based on value in ECX)

invoke ExitProcess**,**0

main endp

end main

When your assembler program compiles and runs, it should print all the numbers in reverse order starting at 13. DON’T OVERTHINK IT 😊