AA2(S3)

Marc Aliaga 16/10/2025

Problem 1

Build an algorithm that counts positive integers less than 1000 that are multiples of eleven. One way to identify whether a number is a multiple of eleven is to add the digits in odd positions on one side and the digits in even positions on the other, and then subtract the two sums so that if the result is 0 or a multiple of 11, the number being analyzed is also a multiple of 11.

Listing 1: Sample C code – Multiples betwen 0 and 1000 of 11.

```c
#include <stdio.h>
#include <stdbool.h>

int main()
{
    int n, contador;
    int centenas, decenas, unidades;
    bool digitos;

    contador = 0;
    n = 1;

    while (n < 1000) {

        if (n < 10) {
            digitos = false;
        }
        else {
            if (n < 100) {
                digitos = (n % 11) == 0;
            }
            else if (n < 1000) {
                centenas = n / 100;
                decenas = (n / 10) % 10;
                unidades = n % 10;
                digitos = ((centenas - decenas + unidades) % 11) == 0;
            }
        }

        if (digitos) {
            contador++;
        }

        n = n + 1;
    }

    printf("Cantidad de multiplos de 11: %d\n", contador);

    return 0;
}
```

Listing 2: Same logic in MASM (x86 assembly) – Multiples betwen 0 and 1000 of 11.

```asm
.386
.model flat, stdcall
.stack 4096
includelib kernel32.lib
ExitProcess PROTO :DWORD
GetStdHandle PROTO :DWORD
WriteConsoleA PROTO :DWORD, :DWORD, :DWORD, :DWORD, :DWORD
STD_OUTPUT_HANDLE EQU -11
.data
bytes_written dd ?
mensaje db "Cantidad de multiplos de 11: ",0
buffer db 16 dup(0)
contador dd 0
handle dd ?
.code
main PROC
push STD_OUTPUT_HANDLE
call GetStdHandle
mov [handle], eax
mov ecx, 1
mov ebx, 0
bucle:
cmp ecx, 1000
jge fin_bucle
mov eax, ecx
cdq
mov esi, 11
div esi
cmp edx, 0
jne no_multiplo
inc ebx
no_multiplo:
```

```
inc ecx
jmp bucle
fin_bucle:
mov [contador], ebx
mov eax, [contador]
lea edi, buffer+15
mov byte ptr [edi], 0
mov ebx, edi
cmp eax, 0
je es_cero
jmp inicio_conv_bucle
es_cero:
dec edi
mov byte ptr [edi], '0'
mov ebx, edi
jmp fin_conv_bucle
inicio_conv_bucle:
conv_bucle:
dec edi
xor edx, edx
mov ecx, 10
div ecx
add dl, "0"
mov [edi], dl
mov ebx, edi
test eax, eax
jnz conv_bucle
fin_conv_bucle:
push 0
push offset bytes_written
push 29
lea eax, [mensaje]
push eax
push dword ptr [handle]
call WriteConsoleA
lea eax, buffer+15
sub eax, ebx
push 0
push offset bytes_written
push eax
push ebx
push dword ptr [handle]
call WriteConsoleA
push 0
call ExitProcess
main ENDP
END main
```

All the explanation of the logic of the MASM code is in the other document.

But in any case, I have done the same program in MASM to be available to see the logic "row by row" and get a deeper understading on whats going on.

PROBLEM 2

Consider the following incomplete C applications. Fill in the spaces provided with code so that they implement the functionality inferred from the comments found within the code itself:

Listing 3: Determinamos cual de los dos es mayor y lo imprimimos.

```

int main() {
    // Declaraci n de variables
    int a, b, mayor;

    // Demandamos los datos al usuario
    printf("Introduce el primer numero:\n");
    scanf("%d", &a);

    printf("Introduce el segundo numero:\n");
    scanf("%d", &b);

    // Determinamos cual de los dos es mayor y lo imprimimos
    if (a == b) {
        printf("Son iguales\n");
    } else {
        mayor = (a > b) ? a : b;
        printf("%d es mayor\n", mayor);
    }

    return 0;
}
```

The cool part of this exercise is the ternary operator, as it lets you write in a single line what would normally take three else if checks.

The operator says: (a > b) ? a : b;, which means that if a is greater than b, then the variable mayor (the greater one) will be a; otherwise, it will be b.

Listing 4: Determinamos cual es la suma de los ‘a’ primeros números y la printamos

```

```

```
//Declaraci n de variables

int a, n, suma;

//Demandamos los datos al usuario

printf("Introduce un numero:\textbackslash{}n");

scanf("\%d",\&a);

//opcion 1:
suma = (int)a * (a + 1) / 2;

//opcion 2
for (n=1; n<= n, n++){
suma += n;
}

printf("La suma de los %d primeros numeros es: %d\n",a,suma);

system("pause");

return 0;

}
```

Listing 5: Determinamos cuales son sus divisores y los printamos.

```

//Declaraci n de variables

int a, n;

//Demandamos los datos al usuario

printf("Introduce un numero:\textbackslash{}n");

scanf("\%d",\&a);

//Determinamos cuales son sus divisores y los printamos

for (n = 1; n <= a; n++) {
        // Si el modulo es 0, es divisor
        if (a % n == 0) {
            printf("%d\n", n);
        }
    }
```