# Chapter 2 Exercises

1. Indicate whether the following statements are syntactically correct or incorrect. If

incorrect, indicate what is wrong with the statement:

1. printf PROTO arg1:Ptr Byte, printlist:VARARG

Correct

1. msg1fmt byte "\n%s%d\n",0

Incorrect – the new line symbol in MASM is represented as **0Ah** (hexadecimal) rather than **\n**. The new line should also be outside the quoted string, separated by a comma.

1. INVOKE printf, ADDR msg1fmt, ADDR number

Incorrect – the **number** variable does not need to be preceded by an **ADDR** address pointer, since it is an integer variable.

1. msg2fmt byte 0Ah,0Ah„"%s",0Ah,0Ah,0

Incorrect – there is an additional comma between the second 0Ah newline and the quoted string. Also, it’s unlikely two newlines are needed before the string.

2. Assuming that the .data section is set up properly, what is wrong with the logic of

the following code segment? How could it be rewritten to avoid the difficulty?

mov num1,5

mov eax,num1

INVOKE printf, ADDR msg1fmt, ADDR msg1, num2

mov num2,eax

Answer: The print statement is invoked before a value is stored in num2, so whatever junk happens to be in that memory location will be printed. Also, **INVOKE printf** destroys the contents of the eax register, so the value finally stored in num2 will be indeterminate. To fix these issues, the “**mov num2,eax”** line should be moved between the “**mov eax,num1”** and “**INVOKE printf”** lines.

3. Given the following MASM program, what will be output to the screen? Be sure to

line everything up properly. Use a lowercase letter b to represent a blank and the words

blank line to represent a blank line:

.386

.model flat, c

.stack 100h

printf PROTO arg1:Ptr Byte, printlist:VARARG

.data

msg1fmt byte "%s%d",0

msg2fmt byte "%s%d",0Ah,0Ah,0Ah,0

msg3fmt byte "%s%d",0Ah,0

msg1 byte "x= ",0

msg2 byte " y= ",0

msg3 byte "z=",0

num1 sdword 1

num2 sdword 2

num3 sdword 3

.code

main proc

INVOKE printf, ADDR msg1fmt, ADDR msg1, num1

INVOKE printf, ADDR msg2fmt, ADDR msg2, num2

INVOKE printf, ADDR msg3fmt, ADDR msg3, num3

ret

main endp

end

Answer:

x=b1by=b2

blank line

blank line

z=3

blank line

4. Given the following MASM program, what will be output to the screen. Be sure to

line everything up properly. Use a lowercase letter b to represent a blank and the

words blank line to represent a blank line:

.386

.model flat, c

.stack 100h

printf PROTO arg1:Ptr Byte, printlist:VARARG

.data

msg1fmt byte 0Ah,"%s%d%s%d%s",0Ah,0

msg2fmt byte 0Ah,"%s%d",0Ah,0Ah,0

msg11 byte "The first number is ",0

msg12 byte ", but the second number is ",0

msg13 byte ",",0

msg2 byte "while the third number is ",0

num1 sdword 5

num2 sdword 7

num3 sdword 11

.code

main proc

INVOKE printf, ADDR msg1fmt, ADDR msg11, num1,

ADDR msg12, num2, ADDR msg13

INVOKE printf, ADDR msg2fmt, ADDR msg2, num3

ret

main endp

end

Answer:

blank line

The first number is 5, but the second number is 7,

blank line

while the third number is 11

blank line

blank line

5. Implement the following C program in MASM. Be sure to use proper spacing on

all output. If necessary, first key in the C program and then implement the MASM

program to insure the MASM program works identically to the C program:

#include <stdio.h>

int main(){

int x, y, z;

x = 1;

y = 2;

z = 3;

printf(%\n%d%s%d%s%d\n\n", x, " + ", y, "= ", z);

return 0;

}

6. Implement the following C program in MASM. Be sure to use proper spacing on all output. If necessary, first key in the C program and then implement the MASM program to insure the MASM program works identically to the C program:

#include <stdio.h>

int main(){

int num1, num2;

printf("\n%s","Enter a value for num1: ");

scanf("%d",&num1);

printf("\n%s","Enter a value for num2: ");

scanf("%d",&num2);

printf("\n%s\n\n","num1 num2");

printf("%s%d%s%d\n\n"," ",num1," ",num2);

return 0;

}

7. Given the following input and output, write both the C and assembly code necessary

to make it look exactly as below. Pay careful attention to spacing and the blank lines:

*Input and Output*

Enter a number: 1

Enter a larger number: 3

Enter an even larger number: 5

1 < 3 < 5

5 > 3 > 1