1.)

a.)

XR=1
ADA+=$101
SBA= $001=01
MULT 01*1=01
Yes it does have the same end result in ACC and memory if they share the same ACC.

b.)

XR=1
Label = 256
LDA+ 255
XR=4
256*255 - 1 * 257= 257
XR = 2
No, it does not end on the same value.

(2) Problem 5.4

a.) Loads XR Register, no flags set
b.) XR Register is loaded, XR Register is subtracted so OV Flag is thrown
c.) XR Register is changed, ACC and AR Registers are changed
d.) XR Register is loaded and subtracted so OV Flag is thrown

3.)

It does not matter if the array is sorted in ascending order, due to it still going thru the AOC formula. You are able to create a SOJ version of the formula since the order does not make a difference in either

(4) Problem 5.10

a.) SOJ gets its addressing mode from the JNO instructions due to it being a combination of SBX and JNO.

b.) The AOC addressing mode comes from the compare instruction because a CMX requires an AOC.

5.)

b.) Nested call: A subroutine called within subroutine calls where the return address is provided by the stack on top/ current return

c.) Stack overflow: A error that occurs when a program tries to write more memory space in the stack than has been allocated.

e.)Indirect Addressing: The addressing value of an operand instruction to determine the address of a memory word

f.) Stack Pointer: small register that stores the address of the last data element added to the stack.

f.) Calling Convention: Describes the rules for accessing data between the caller and the subroutine

g.) Stack Frame: Changes the address computation so that the content of the FP is added to the addressing value before compilation is complete

(6) Problem 6.2

CBA is call by addressing, and copies the addresses of the parameter, It is used when the caller's actual parameter needs to be changed, such as passing arrays. CBV is call by value, and should be used when sub-routines are not used to return data, used for small data structures.

7.)

Formal: Private names for variables that are pushed from calling the program

Actual: When a subroutine call is complete, the caller provides its own variables to the subroutine

(8) Problem 6.5

Saving registers is important to prevent the subroutine from damaging the content of the registers that the program calls/uses. The two saving conventions are, 1.) the caller saves the critical registers, and the second method is the subroutine saves the registers.

9.)

```
procedure Batman(a:integer_array; b:integer)
function Robin(var c,d: integer; e,f:integer_array; g:integer):integer;

Batman:
    Var
        a:integer_array
        b:integer
Robin:
    Var
    c,d,g : integer
    e,f: integer_array
```

(10) Problem 6.8 using the above information

```
Main:
LDS# $E00
LDF# $000
PSH# a
PSH# b
JSR Robin
ADS# 2
HLT
```

```
a:      .WORD 0
b:      .BLKW 10
RTN
.END
```

11.)

```
        Robin: BGN# 000
                LDS# F00
                LDF# $000
                PSH# c
                PSH# d
                LDA e
                LDA# f
                STA e
                STA# f
                LDX g
                STX g
.EQU @,200
        .EQU c,0
        .EQU d,0
        .EQU e,0
        .EQU f,0
        .EQU g,0
```