2-15) The value of M when by:

i) reference is 2

ii) value-result is 3.

Explanation:

By-reference: X and y are pointing to N at the callee AR. Hence any change of value to either one of them (X or Y) results in an immediate change to the value of N, and consequently to the other formal value. If we change X, N will change and hence Y, and vise versa.
X= 1 in SUB TEST changes the value of N at the main and Y at TEST to be 1. Hence, adding X to Y will result in 2 which will be stored in M (pointed to be Z). Hence M will end up having the value 2.

By-value-result: inside TEST, the execution of X=1 does not change N, hence Y remains 2. When we do Z = X + Y we are adding 1 + 2 and will end up having the value 3 upon existing the TEST subroutine.

2-16)

i) If the addresses are computed once, on entry at the caller, then the program prints 10, 20, 11.

Steps:

1) At the caller side: "Call SUB (I, A(I))"
the value of the actual parameters: I is 1 and A(1) = 10

2) at the callee side (inside SUB):
The formal actual substitution process will place the value 1 (of the actual I) into its corresponding formal K, which will have 1, and the formal X will have 10 (the value of the actual A(1)).
"PRINT X" will output 10
After executing the next two lines:
K will have 2 and X will have 20
Then we his return:
At the caller side: the actual I will have 2, and
A(1) will have 20.
"PRINT A(1), A(2)" will output 20, 11

ii) If the addresses are computed on entry and exit, but the obtained results from the formal K is stored back in "I" before re-computing the address of the I^{th} element of A [the actual A(I)], then the program prints 10, 10, 20.

Steps:

1) At the caller side: "Call SUB (I, A(I))"
   the value of the actual parameters: I is 1 and A(1) = 10

2) at the callee side (inside SUB):
   The formal actual substitution process will place the value 1 (I) into its corresponding actual K will have 1, and X will have 10 (A(1)).
   "PRINT X" will output 10
   After executing the next two lines:
   K will have 2 and X will have 20
   Then we his return:
   At the caller side: the actual I will have 2, and
   A(I) which is A(2) will have 20, where
   A(1) will not be affected [remains 10]
   since I is 2 now!
   "PRINT A(1), A(2)" will output 10, 20

iii) If the addresses are computed on entry and exit, but the address of the I^{th} element of A is computed before storing the results back in the actual parameters: I and A(I); then the program prints 10, 20, 11 (same as in (i) above).

Steps:

1) At the caller side: "Call SUB (I, A(I))"
   the value of the actual parameters: I is 1 and A(1) = 10

2) at the callee side (inside SUB):
   The formal actual substitution process will place the value 1 (I) into its corresponding actual K will have 1, and X will have 10 (A(1)).
   "PRINT X" will output 10
   After executing the next two lines:
   K will have 2 and X will have 20
   Then we his return:
   At the caller side: the address of A(I) is computed with I still 1 (we did not copy the result of 2 back into I yet), i.e., A(1) gets 20 on it, where A(2) still 11.
   "PRINT A(1), A(2)" will output 20, 11