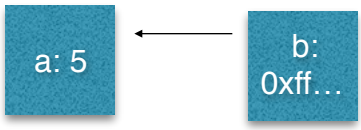
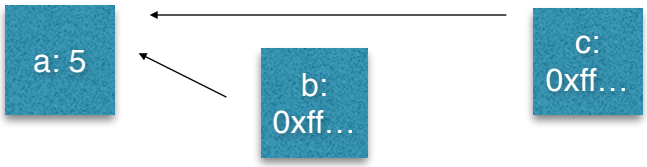
1. Passing an integer as a parameter copy's the value to the functions context, and any changes to the value inside the function need to be return'd out, and it's the callers responsibility to use those values. A reference passes the memory address where the value of the variable is stored, a pointer passes the memory address of the variable. Both a pointer and a reference allow direct modification (mutation) of the value of the variable.
2. Assuming data is contiguous starting at the 0th index. valid data is stored up to the first null terminator (\0) or the 50th index. removing data would entail adding a null terminator to the point where the data should end (likely the 0th index).
3.
    a) no, you you just keep track of the largest number yet entered and if a larger one is encountered set the variable to be the new larger number.
    b) no, you just need to keep a sum of the numbers entered and a count of the numbers entered. the average would be sum/count
    c) yes! In this case the simplest, or really the best, way would be with a dynamically allocated array.
4. Nearly everything is stored on the stack. stack memory is automatically free'd when it falls out of context (is popped off). heap memory is anything allocated with the new keyword. heap memory is freed either with the delete keyword or when the program exits.
5.
    1. a is set to the value 5

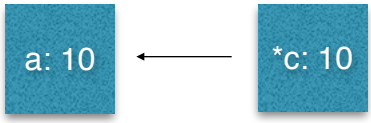
    2. b points to a



    3. c points to a (b passes a's memory address to c)



    4. c is dereferenced and set to 10 (aka the thing c is pointing at (a) is set to 10)



    5. a is set to 3

6. a is 3