2.

a) Does this hash function satisfy any of the requirements for a crypto-hash function listed below? Explain your answer:

* variable input size
  + This function satisfies this requirement because the input size of M can be any positive integer.
* fixed output size
  + Every case will have the same size because at the end whatever the sum is, the result is going to be a number between 0 and n - 1. So it will require the number of bits that n requires to be represented.
* efficiency (time-space complexity)
  + This will be linear time because in one single pass all the elements will be added. And mod is a linear operation so timewise it is linear. As of space-wise, it is constant space.
* first and second pre-image resistance
  + First: Do not satisfy. This is because if I choose two random messages it could be possible for them to have the same result because addition is commutative.
  + Second: Do not satisfy. This is because we can find numbers such as 1 to 10 and 10 to 1 that will produce the same result because addition is commutative.
* strong collision resistance
  + Do not satisfy. This is because we can find numbers such as 1 to 10 and 10 to 1 that will produce the same result because addition is commutative.
* pseudo-randomness (unpredictability of the output)
  + Not satisfy. This is because if I change the order of the numbers for example, I have numbers 1 to 5, and I change the order to be 1 2 3 5 4, I should get a different output, but I get the same because addition is commutative and I can predict it. Or if I am adding numbers 10 to 20 and then 10 to 19 and instead of 20, I add 21 at the end, the hash difference will be of 1 because I am just adding and doing the mod operation to that addition.

b) Repeat part (a) for the following hash function:

* variable input size
  + This function satisfies this requirement because the input size of M can be any positive integer.
* fixed output size
  + Every case will have the same size because at the end whatever the sum is, the result is going to be a number between 0 and n - 1. So it will require the number of bits that n requires to be represented.
* efficiency (time-space complexity)
  + This will be linear time because in one single pass all the elements will be added. And mod is a linear operation so timewise it is linear. As of space-wise, it is constant space.
* first and second pre-image resistance
  + First: Do not satisfy. This is because if I choose two random messages it could be possible for them to have the same result because addition is commutative.
  + Second: Do not satisfy. This is because we can find numbers such as 1 to 10 and 10 to 1 that will produce the same result because addition is commutative.
* strong collision resistance
  + Do not satisfy. This is because we can find numbers such as 1 to 10 and 10 to 1 that will produce the same result because addition is commutative.
* pseudo-randomness (unpredictability of the output)
  + Not satisfy. This is because if I change the order of the numbers for example, I have numbers 6 to 10, and I change the order to be 6 7 9 8 10, I should get a different output, but I get the same because addition is commutative and I can predict it. Or if I am adding numbers 1 to 10 and then 9 to 1 and instead of 10, I add 11 at the beginning, the hash difference will be of 21 because I am just adding and doing the mod operation to that addition.

c) Calculate the hash function of part (b) for M = (189, 632, 900, 722, 349) and n = 989.

Answer: 229