How your hash function works: its complexity, how you came up with the idea for the implementation of your hash function, how you tweaked/improved your original idea, etc.

The hash function works in a simple manner first of all it takes 2 inputs one being a string and another being an int. Then the function is going to initialize a variable hash and give it the value of zero. Then it is going to run a for loop for each J number in the string Data. The for loop then initializes a variable x for the first character in the string after that it gives the hash a new value which is multiplied by 786 and then squared after that The value of x is added to it. The for loop will repeat this process for each index of the String. After the for loop, it gives the absolute value of the hash MOD size. The original idea is just the regular hash function from the slides although I improved it by updating the hash as hash = (786 * hash)^2 +x; so that there are fewer collisions. (NOTE: There is no special reason for the number being 786 it could be replaced by 5 and the results would be almost the same.)

Its performance in the four tests: how well did the hash function do at distributing indexes, How well did it avoid collisions, what was the ratio of successful hashes to collisions, etc.

For the unimproved hashfuctions the first test had a collision rate of ~56%, for the second one it was ~89%, the third one it was ~95% and the fourth one was 92%. My assumption would be because the hash variable not having too many options it kept repeating itself.

100 randomly generated J Numbers hashed for an array with a length of 127 had
Collision total = 71
500 randomly generated J Numbers hashed for an array with a length of 512 had
Collision total = 457
1000 randomly generated J Numbers hashed for an array with a length of 1024 had
Collision total = 955
25000 randomly generated J Numbers hashed for an array with a length of 27000 had
Collision total = 24943

How your hash function performed compared to the simple string hash function

The custom function had better results as for the first test the collision rate was ~24%, the second test was ~33%, the third one was ~34% and the fourth one was 32%. So with the improved function, the collision rate was almost decreased by 50% with the first test and by around 66% with the rest.

100 randomly generated J Numbers hashed for an array with a length of 127 had
Collision total = 30
500 randomly generated J Numbers hashed for an array with a length of 512 had
Collision total = 167
1000 randomly generated J Numbers hashed for an array with a length of 1024 had
Collision total = 344
25000 randomly generated J Numbers hashed for an array with a length of 27000 had
Collision total = 8712