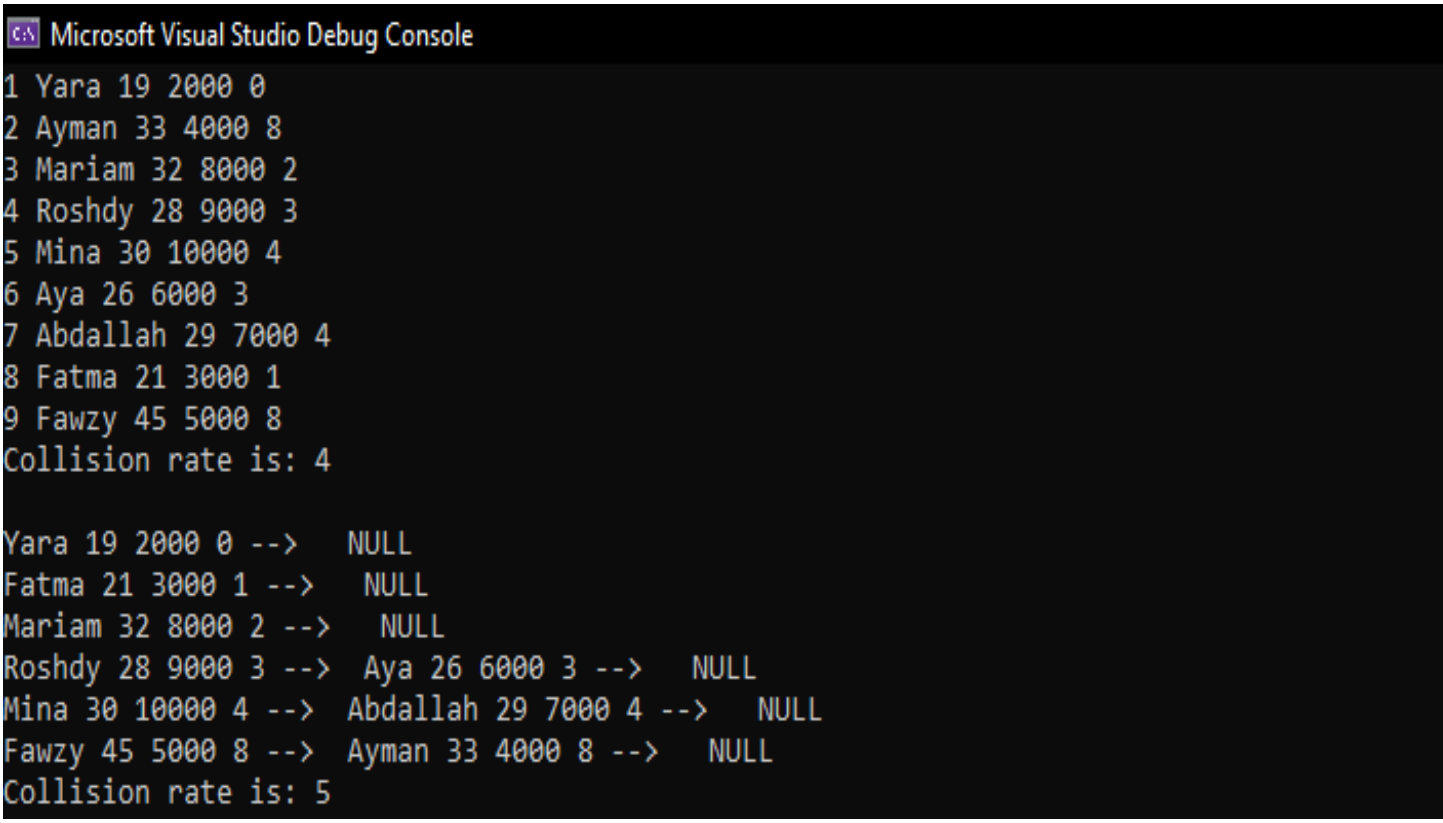
Assignment (4)

Report



The first example is using Linear probing and a dynamic array. My hash function used was depending on the years of experience% the size which was 9 in the example and can be changed through the default constructor. The reason for choosing the years or experience and not the salary will most likely have a 0 at the end and so they will all collide. Why the experience and not the age, well basically this was by choice but they will mostly be the same, but the order will not be the same. Also, I chose experience as it was mostly just a single digit.

The collision rate was calculated through this formula:

Collison rate is $C = M * (M-1) / 2T$

where M is the number of elements to be hashed

T is the total number of hashed elements.

According to this website: https://iq.opengenus.org/probability-of-collision-in-hash/#:~:text=of%20hash%20values.-,If%20we%20hash%20M%20values%20and%20total%20possible%20hash%20values,(M%2D1)%20%2F%202T

The second example is using LinkedList and chaining My hash function used was depending on the years of experience% the size which was 10. The reason I chose 10 is that so it could function on any number of employees. The reason for choosing the hash function to be the same as the first example is still the same.

The collision rate was calculated through the same formula but with size 10.

I believe that the LinkedList is better as it handles collisions better and will never return that the hash table is full as there will always be a place for any element. The only downside to it is that it leaves gaps in the HashTable. Other than that, it is more efficient.