**Data Structures Lab 9 Report**

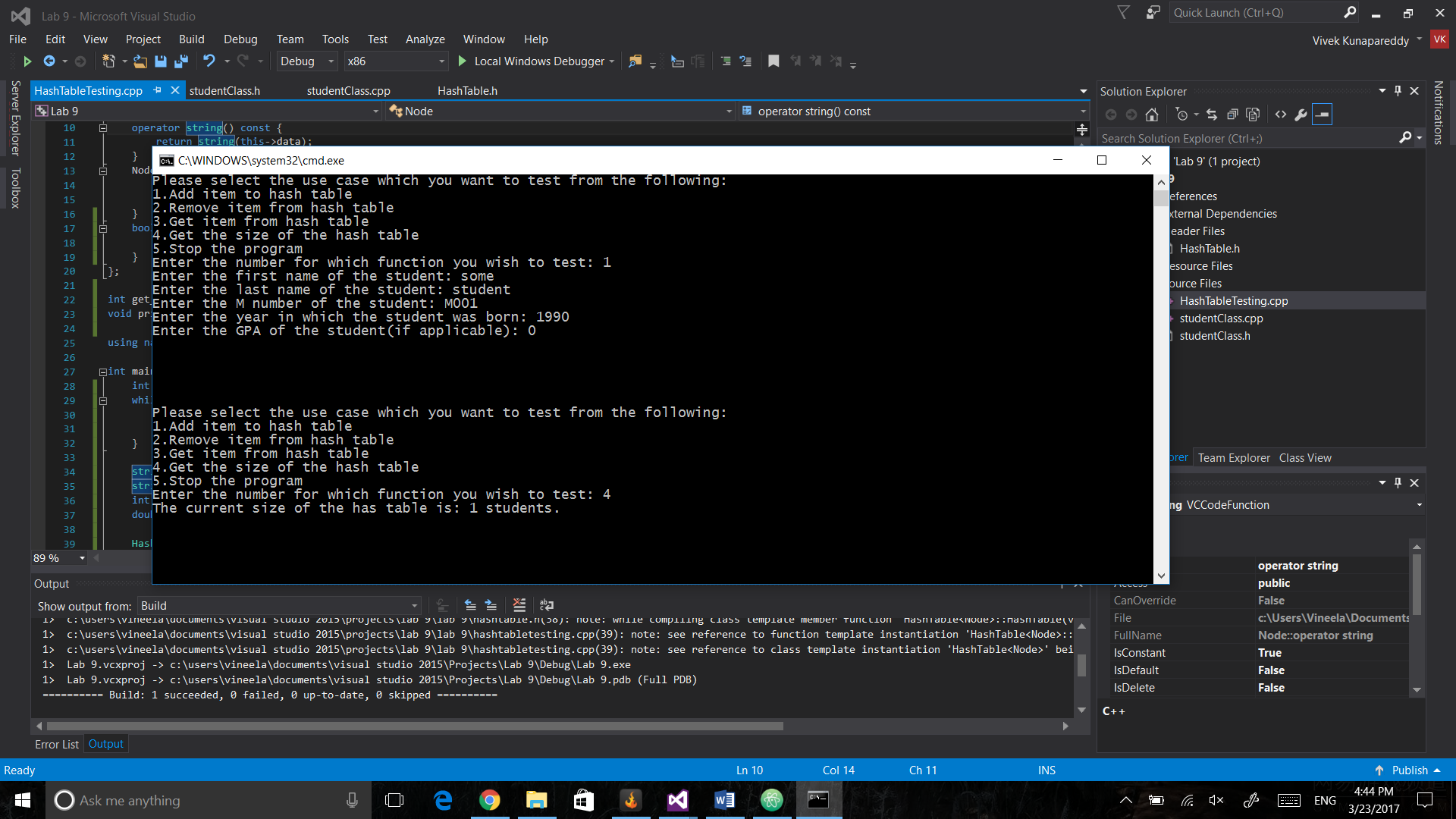
Group Members: Vivek Kunapareddy, Yuan Cheng

**Discussion of objectives/concepts:**

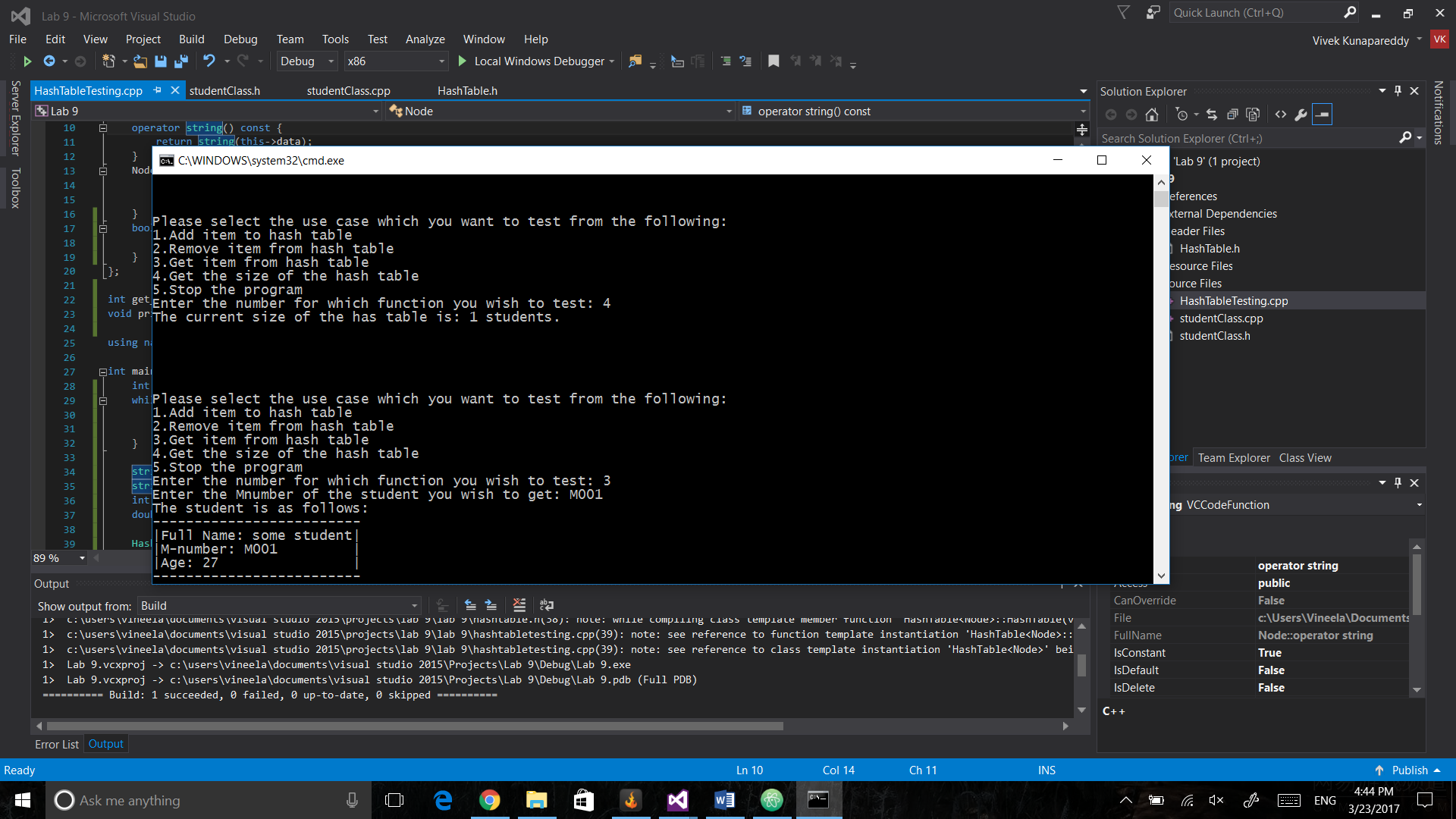
This assignment involved the creation of Hash tables using two different implementations. The two implementations explored were linear probing and chains(linked lists). These concepts are going to be important as we transition into the software development workforce as hash tables are a widely used data structure and implementing it allows us to understand how they function behind the scenes.

**Task 3 Screenshots:**

Testing adding a student and removing a student who doesn’t exist:

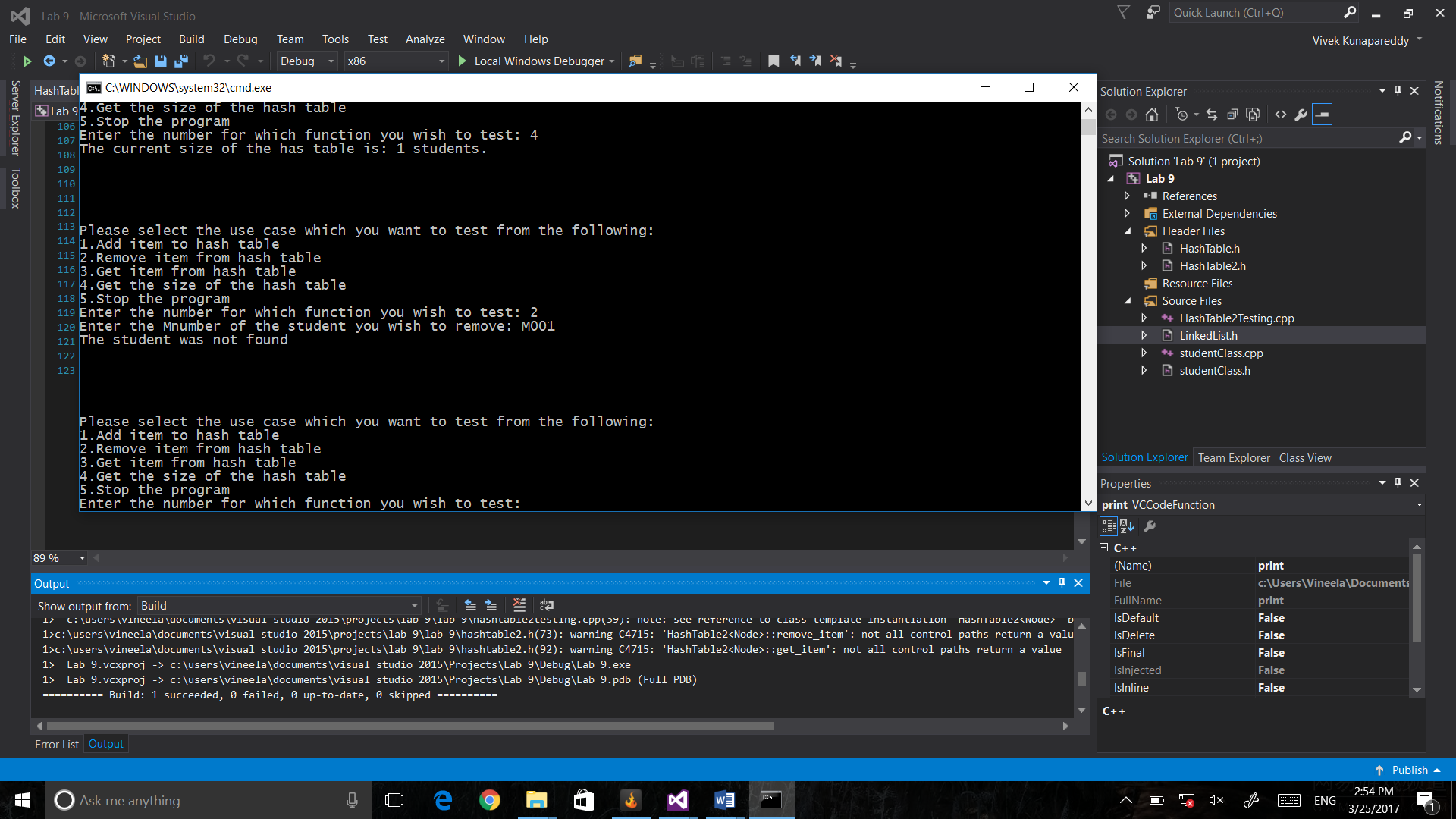


Testing size and removing an existing student:

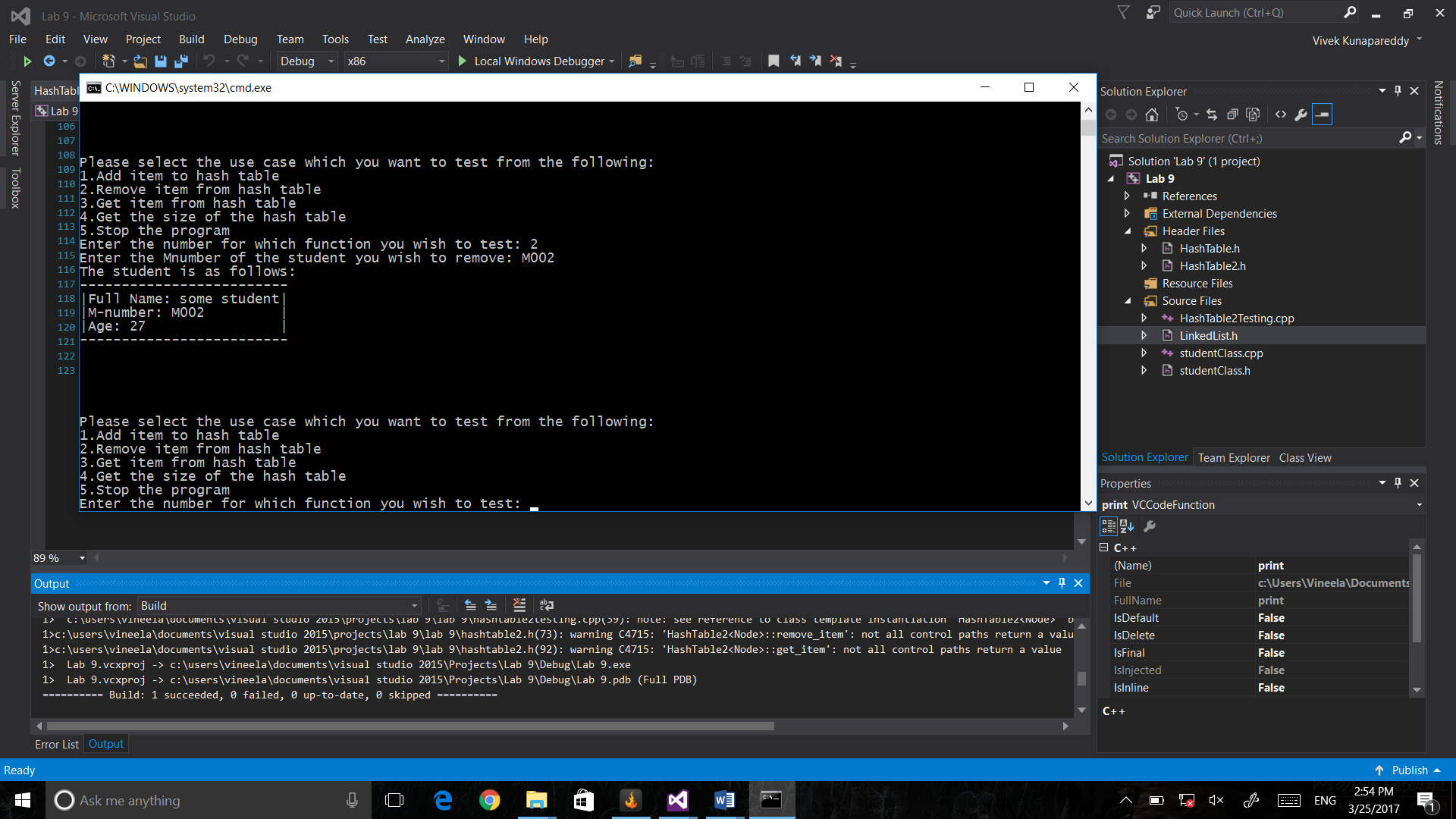


**Task 4 Screenshots:**

Testing size and removing a student who doesn’t exist:



Testing removing a student who exists:



**Task 5 Table and Discussion:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size-> | 100 | 150 | 200 | 250 |
| Linear probing | 26 | 5 | 36 | 2 |
| Chains | 2 | 1 | 3 | 1 |

Clearly as we follow the trends in the table, chaining usually has a lesser number of operations compared to linear probing. However, we can also see that as the size allotted to the hash table increases, the number of operations start normalizing and converging towards the same value. This implies that there must be a breakthrough point at which, given a set number of values to be hashed, we can say that a hash table of a multiple N times the set number will have the same number of operations regardless of implementation. This can be investigated by trying to keep increasing the size allotted to the table and seeing at what point both the values are similar.

**Modification for Task 2, 3, 4:**

Task 2 required adding a string() operator overload to the student class

Task 3 required reducing the inputs given to the user to match the number of methods present in the Hash Table class and also modifying the program to run with Hash Tables instead of linked lists.

Task 4 required overriding multiple functions of the HashTable class to match the linked list implementation. This included get\_item, remove\_item, add\_item, constructors and destructors.

**Group contributions:**

The hash table class was done by Vivek,

The student class redesign was done by Yuan.

The rest of the programming was done together.

**Compilation Instructions:**

Make sure to only have HashTableTesting.cpp or HashTable2Testing.cpp or PerformaceTesting.cpp when compiling to avoid having multiple main functions.

When asked for input, please only enter the numbers in the shown range.