17.05.2020

CS202 Homework4

Irmak Akyeli

21803690

Sec: 02

Bilkent University
CS

Part 1)

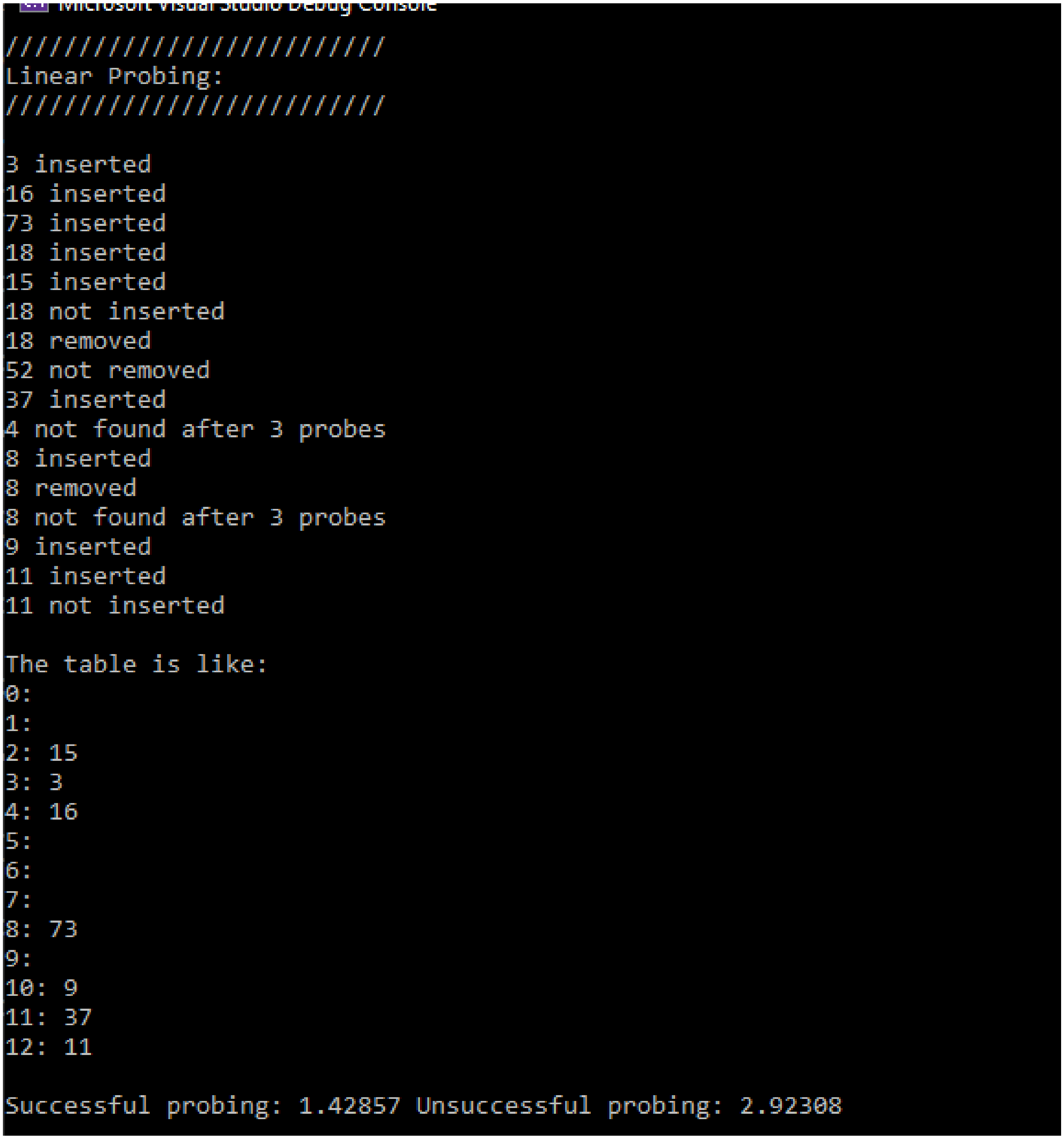
I implemented the HashTable with one big function findIndex, which takes several control inputs like toBeSelected and like toBeInserted. It first checks the collision resolution strategies. In an if statement it determines the operation to be used which was stored previously in the constructor and then looks for the index which the item should be placed. Each strategy has a condition where they stop if a counter set reaches the table size. All other functions asked to be implemented like search, insert or remove calls to findIndex function with the control signals. toBeSelected signal lets the function to search for the given item and when it is false it looks for an appropriate gap to put the item. toBeInserted lets you instert an item to an already removed location. It also checks several conditions like the item given is already in the table or is 0 which will create an error in Double hashing. The hashtable uses a Hashnode which has 3 bool properties: isFull, isFree and isRemoved. Each shows a different condition and helps the function.

Part 2)

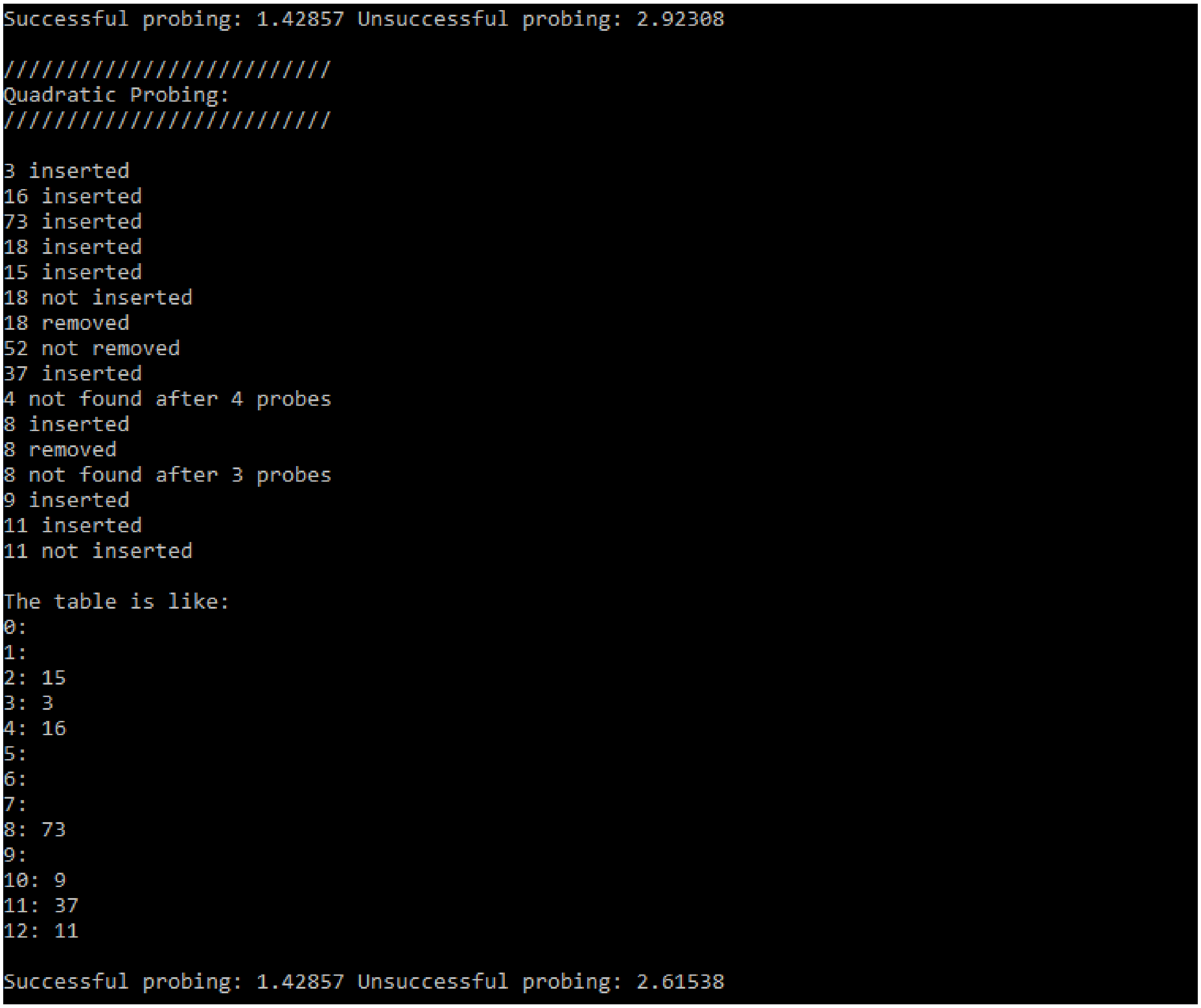
I used a table with size 13 and with the following input text file:

```
I 3

I 16

I 73

I 18

I 15

I 18

R 18

R 52

I 37

S 4

I 8

R 8

S 8

I 9

I 11
```

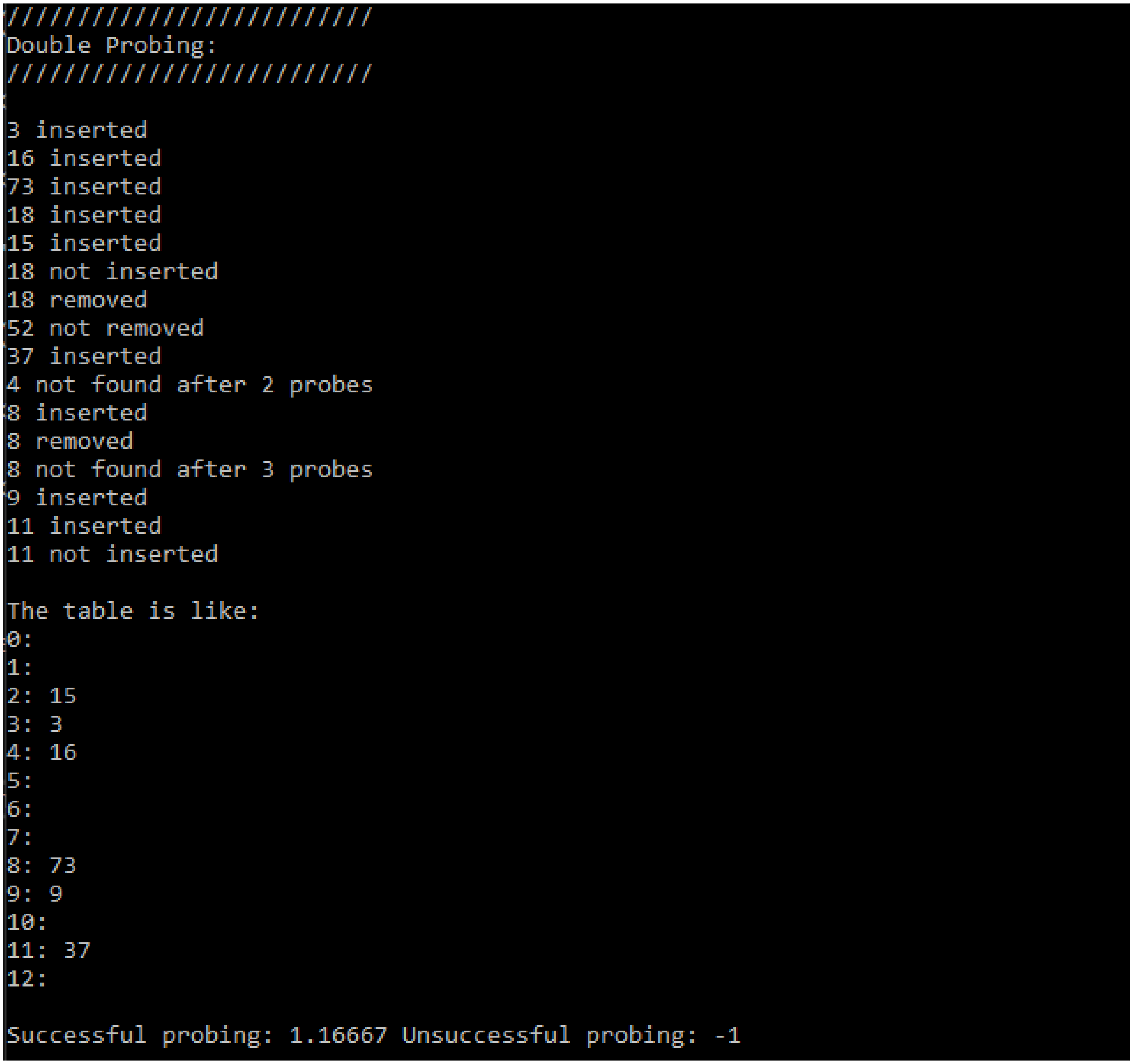
And obtained the following results:



```
//////////////////////////
Linear Probing:
//////////////////////////

3 inserted
16 inserted
73 inserted
18 inserted
15 inserted
18 not inserted
18 removed
52 not removed
37 inserted
4 not found after 3 probes
8 inserted
8 removed
8 not found after 3 probes
9 inserted
11 inserted
11 not inserted

The table is like:
0:
1:
2: 15
3: 3
4: 16
5:
6:
7:
8: 73
9:
10: 9
11: 37
12: 11

Successful probing: 1.42857 Unsuccessful probing: 2.92308
```



```
Successful probing: 1.42857 Unsuccessful probing: 2.92308

//////////////////////////
Quadratic Probing:
//////////////////////////

3 inserted
16 inserted
73 inserted
18 inserted
15 inserted
18 not inserted
18 removed
52 not removed
37 inserted
4 not found after 4 probes
8 inserted
8 removed
8 not found after 3 probes
9 inserted
11 inserted
11 not inserted

The table is like:
0:
1:
2: 15
3: 3
4: 16
5:
6:
7:
8: 73
9:
10: 9
11: 37
12: 11

Successful probing: 1.42857 Unsuccessful probing: 2.61538
```



```
//////////////////////////
Double Probing:
//////////////////////////

3 inserted
16 inserted
73 inserted
18 inserted
15 inserted
18 not inserted
18 removed
52 not removed
37 inserted
4 not found after 2 probes
8 inserted
8 removed
8 not found after 3 probes
9 inserted
11 inserted
11 not inserted

The table is like:
0:
1:
2: 15
3: 3
4: 16
5:
6:
7:
8: 73
9: 9
10:
11: 37
12:

Successful probing: 1.16667 Unsuccessful probing: -1
```

Part 3)

For Linear Probing:

In successful search: 1.42857 which should be 1.58333 by theorical

In unsuccessful search: 2.92308 which should be 2.8474 by theorical

For Quadratic Probing:

In successful search: 1.42857 which should be 1.4359 by theorical

In unsuccessful search: 2.61538 which should be 2.16666 by theorical

For Double Probing:

In successful search: 1.16667 which should be 1.4359 by theorical

In unsuccessful search: N/A which is also N/A in theorical results.

All the theorical results are calculated using the given formulas from the cs202 hash table slides and as the values does not act or cumulate as predicted, we see slight changes in the experimental and theorical results. One important aspect of this difference is the number of the table size and the inputs which means as the theorical results calculate the average from a sufficiently large hash table, our results will be more similar if we take a larger hash table with more item inside. Yet as we take a relatively small table with few inputs, the cumulation is not as random as these formulas are calculating, therefore we obtain a scope of error.