

# Assignment 9 Technical Report

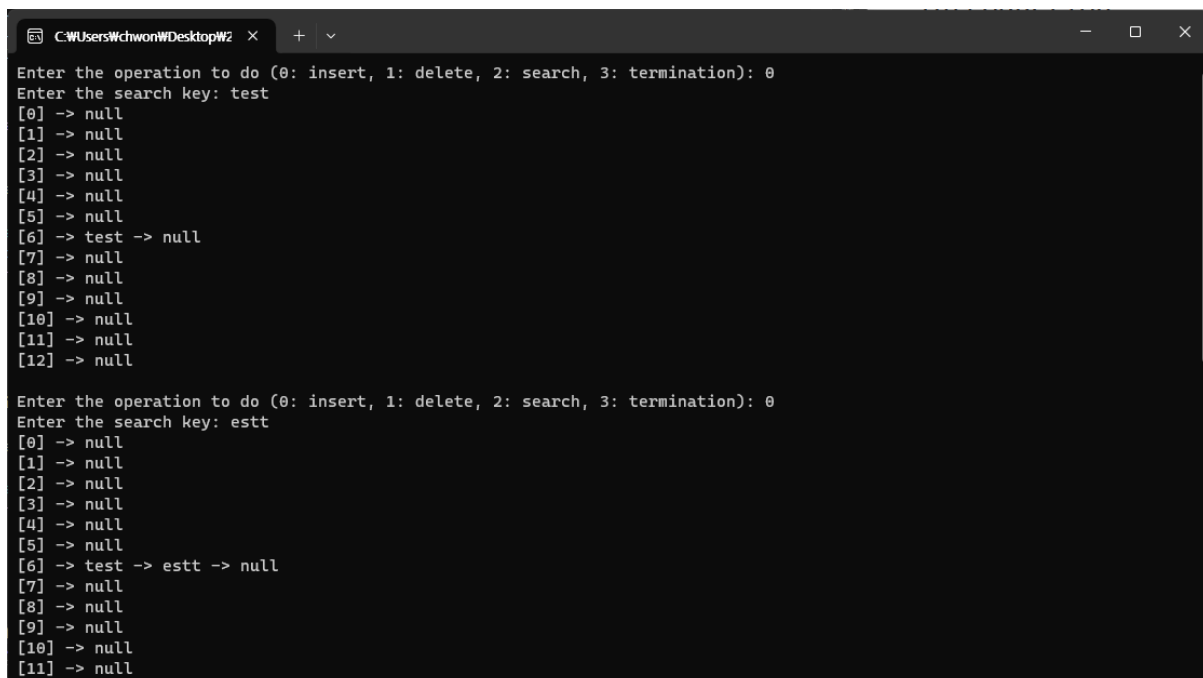
2171039 Chaewon Lee

## # Code 1

This code is actually same as the code that is included in the lecture file, but it includes new delete function for the chaining.

First, in main function, it gets what operation to do for input. Then, if you press 1, it operates deletion operation. Hash\_chain\_delete function, which does this deletion, first checks whether the given key is inside the hash table or not. And, if it is not found, then it prints item is not inside the table, when it found the item, first It should check where the item is inside the table. When it is the first item of that hash value, which means there is no parent node for that item, the next item to the given item should replace the position of that first spot. And when it is not, then parent node of the given item links the item which was the link of the given item, and set the link of the given item as NULL. After this process is completed well, function prints that item deletion is succeeded. You can check this by the full hash table that is automatically printed after every operation.

## Result



```
C:\Users\Wchwon\Desktop\W2 x + -
Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: test
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: estt
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> estt -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
[11] -> null
```

```
C:\Users\Wchwon\Desktop\W2 x + v
Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 0
Enter the search key: tset
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> test -> estt -> tset -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: test
Item deletion success.
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> estt -> tset -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
```

```
Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: tset
Item deletion success.
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> estt -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination):
```

```
C:\Users\Wchwon\Desktop\W2 x + v
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: estt
Item deletion success.
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null

Enter the operation to do (0: insert, 1: delete, 2: search, 3: termination): 1
Enter the search key: estt
Item not found.
[0] -> null
[1] -> null
[2] -> null
[3] -> null
[4] -> null
[5] -> null
[6] -> null
[7] -> null
[8] -> null
[9] -> null
[10] -> null
[11] -> null
[12] -> null
```

You can see the deletion of the given item is going well.

## # Code 2

The code generates random numbers, and then inserts those numbers inside the binary search tree. Then, it prints the numbers in sorted order with inorder traversal.

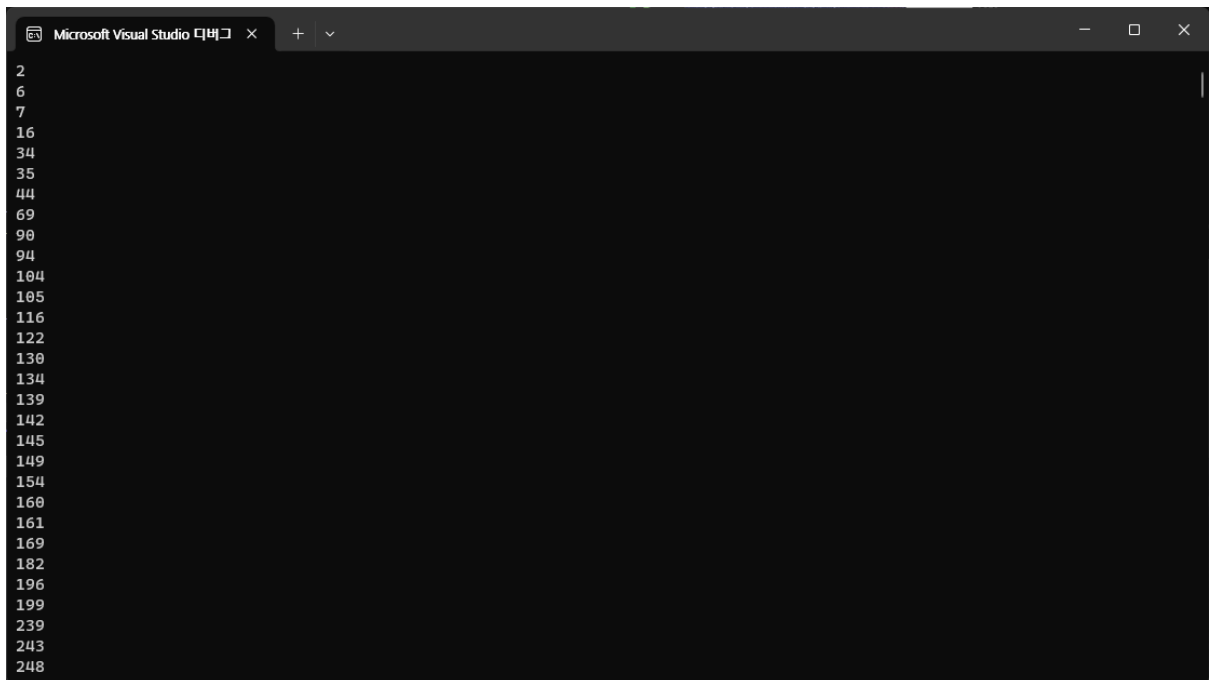
First, in main function, input array is created and new numbers are inserted into the array for 1000 times, which is the size of the array. During this operation, it checks whether the number is already inserted into the array or not, to get the result that all numbers inside bst have distinct values. To do this, when the number is used, the code checks that area of used array as true, and while picking the number, when the number selected's value of used array is true, it picks again until number never used comes out.

And while this puts the value inside the array, it also insert the value inside binary search tree with `bst_insert` function. This first checks whether the given root node is NULL or not. If it is null, this means that the data is the first element of the tree, so it sets the root node's data as that data. When it is not, it traverses until there is no child node left. If the number of the node is bigger than the key, it moves to left node. When it is not, it moves to right node. When it meets the end of the tree, the data is inserted next to the node that is one level higher. If the data is smaller, then it is inserted to the left. When it is not, it is inserted to the right.

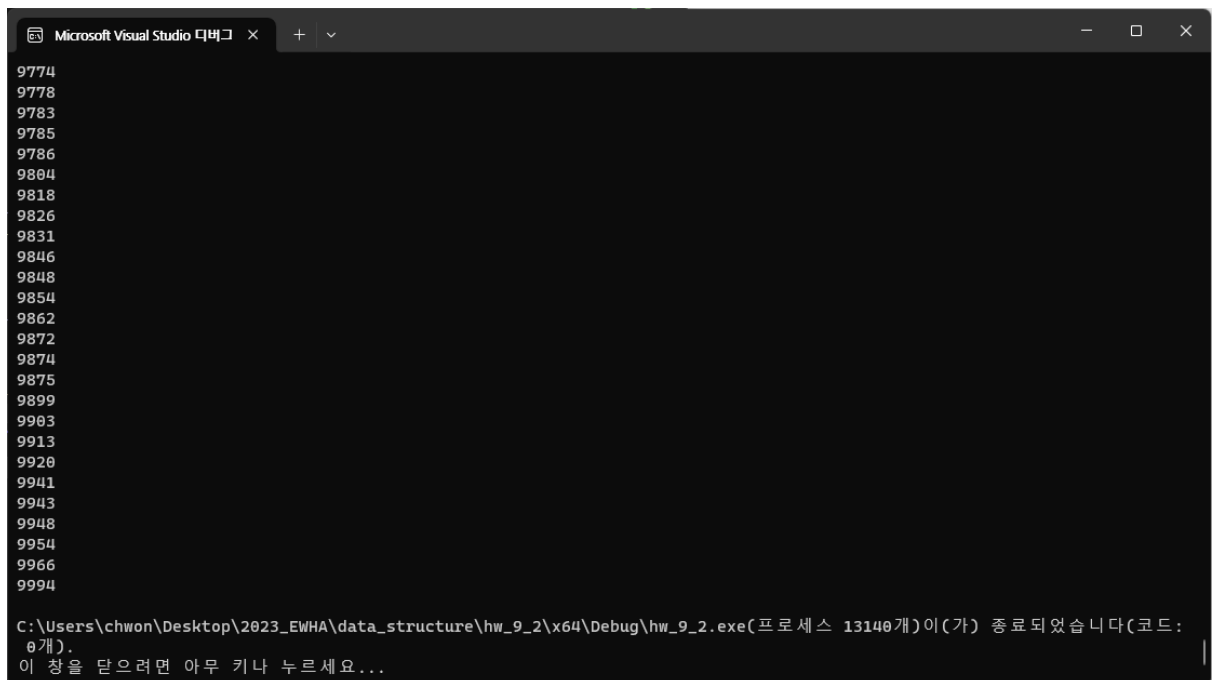
Finally, `bst_print` function in order to traverse the tree and prints the result recursively. It first calls the `bst_print` function for the left node and prints the value in the root node. Then it calls the same function for the right node. If the node that is sent as an

argument is NULL, it returns.

## Result



```
Microsoft Visual Studio 디버그 콘솔
2
6
7
16
34
35
44
69
98
94
104
105
116
122
130
134
139
142
145
149
154
160
161
169
182
196
199
239
243
248
```



```
Microsoft Visual Studio 디버그 콘솔
9774
9778
9783
9785
9786
9804
9818
9826
9831
9846
9848
9854
9862
9872
9874
9875
9899
9903
9913
9920
9941
9943
9948
9954
9966
9994

C:\Users\chwon\Desktop\2023_EWHA\data_structure\hw_9_2\x64\Debug\hw_9_2.exe(프로세스 13140개)이(가) 종료되었습니다(코드: 0x0).
이 창을 닫으려면 아무 키나 누르세요...
```

You can see all results are printed in increasing order well.