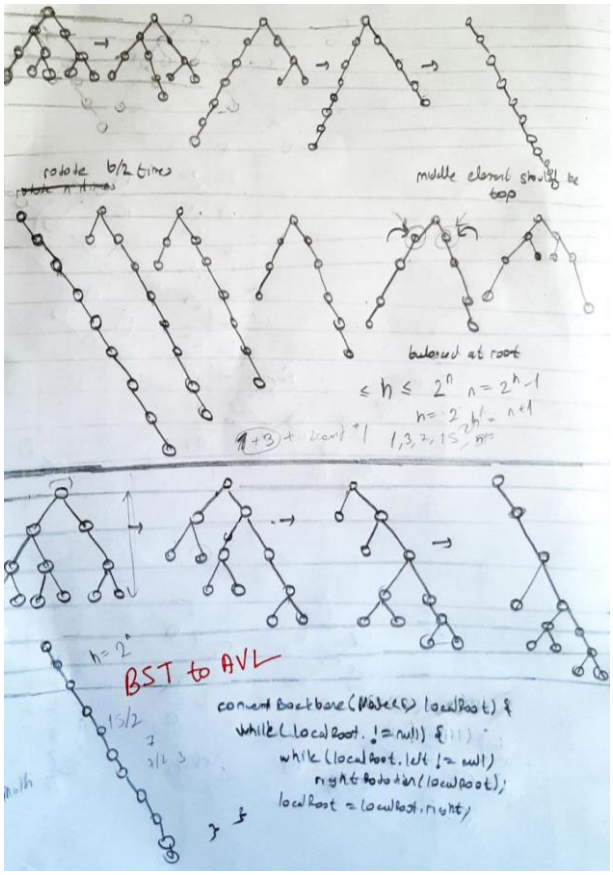
GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 7 Report

Emirkan Burak Yılmaz
1901042659



1. SYSTEM REQUIREMENTS

Q1.

- A method which generates a binary search tree (BST).
- Method takes a binary tree and an array of items as input, and it returns a binary search tree as output. Both binary tree and array contains n items and each item in the array is unique.
- Method should create a BST which has same structure with the binary tree and contains all the items in the array.

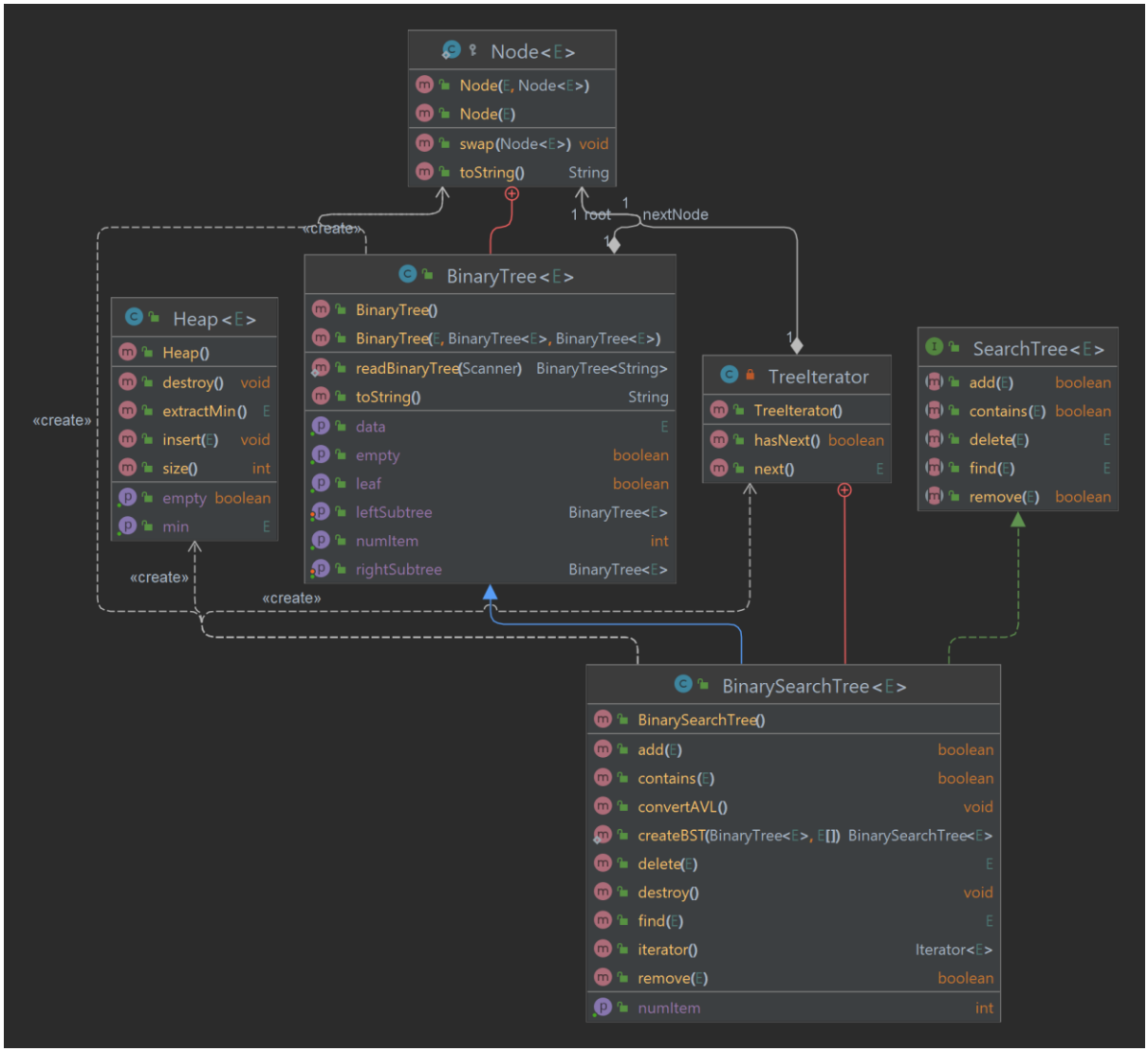
Q2.

- A method converts a binary search tree (BST) to AVL tree.
- Method takes a BST and returns the AVL tree obtained by rearranging the BST.
- Method should rearrange the BST by using rotation operation.

Q3.

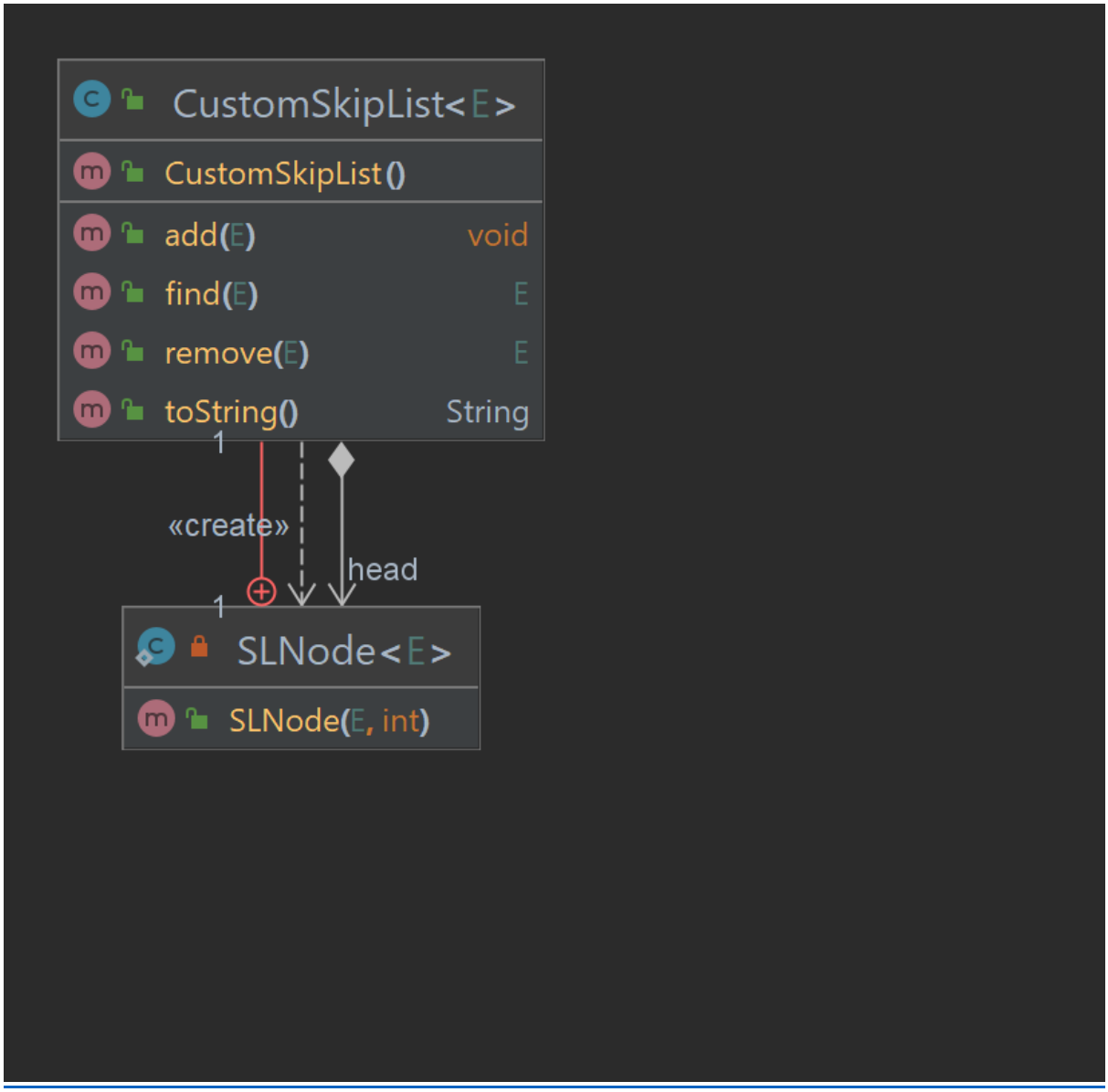
- A new version of the skip list data structure, CustomSkipList.
- The CustomSkipList has two level lists in default.
- A new level list is added to the skip list each time when the size of the first level list reaches multiples of 10.
- When a new level is added to the skip list, the tall items (contained in the more than one level) are appended to a one-level upper list.
- An item can be also appended to upper-level lists during insertion operation.
- The item is always appended to the first level list, and it is appended to higher levels by increasing chance with the number of items in the range between its closest tall neighbors on the left side and the right side. The probability of appending an item to a higher level is calculated by dividing the number of items between two tall neighbors by 10.

2. CLASS DIAGRAMS

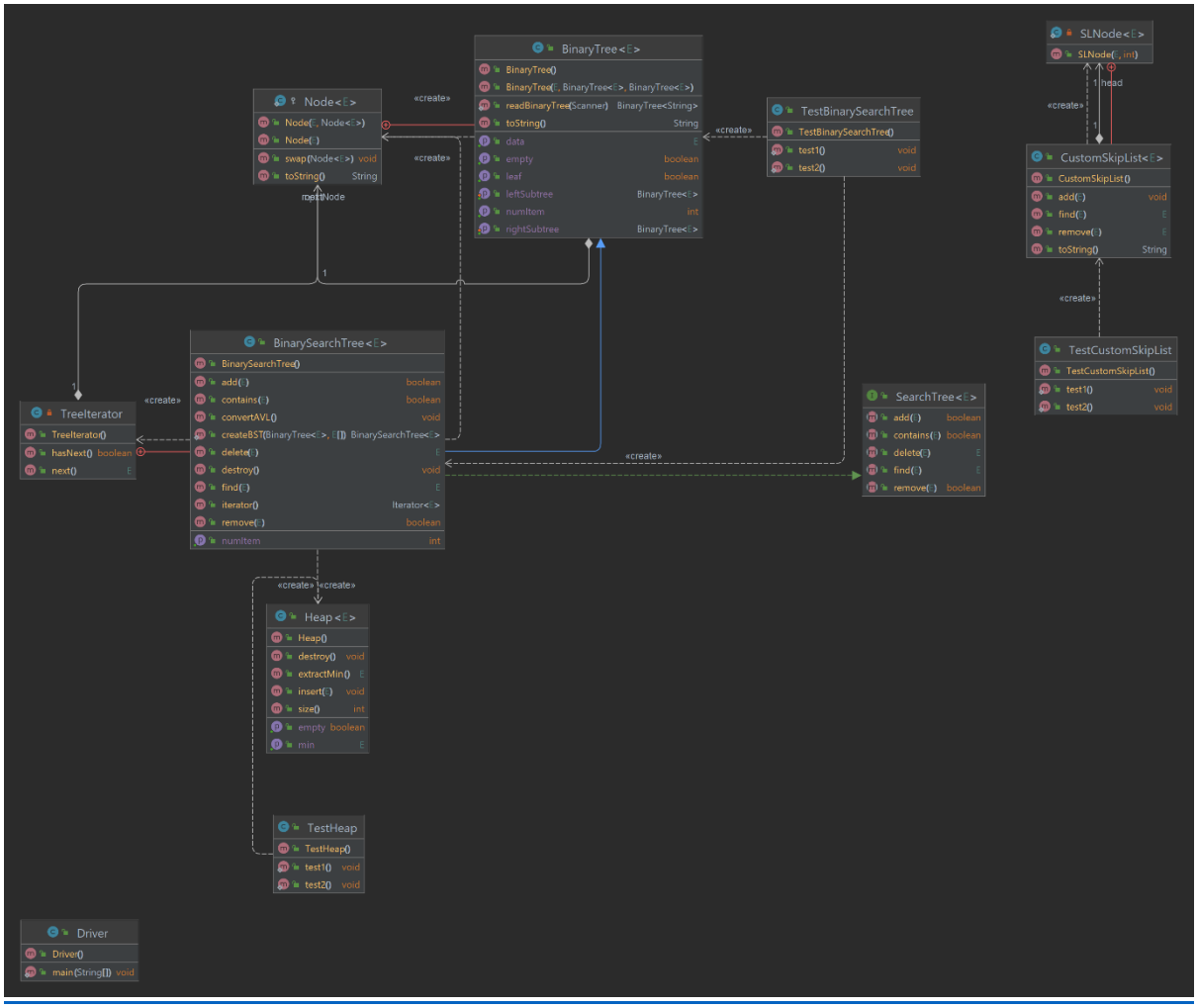


[BinarySearchTree Class Diagram]

[(convertAVL – createBST)]



CustomSkipList Class Diagram



[All the Project Class Diagram]

3. PROBLEM SOLUTION APPROACH

Q1. First create an min-heap from the array of items to access minimum item to reach the minimum item easily. To construct same structure of binary tree, in-order traverse the binary tree. During traversal create and link the nodes of the BST according to structure of binary tree. If the traversing node is non-null, then create a new node for the BST. Each created node gets the value from the heap. First extraction on heap gives the smallest value on the array and it happens at most left child. Similarly last extraction on heap gives the largest value on the array and it happens at most right child. So, extracting the values of heap and assigning them to new created nodes is enabled to maintain the BST order property. As a result, the formed tree has same structure with the given binary tree and contains the items in the given array.

Q2. First, I realized that it is easy to convert a skewed tree to AVL tree rather than randomly structured binary tree. Any tree can be easily converted to skewed structure. After getting skewed structure, start the root of skewed tree and rotate left or right depends on the skewed side. Apply rotation around the root till both sides have almost same height. Then continue to balance recursively the left and right subtrees with same approach.

After some research, I found that there is better algorithm called Day-Stout-Warren (DSW). Since I found it better than my, I try to understand and implement it. The algorithm takes linear time without using auxiliary space. Algorithms consist of two steps. First step is converting BST into a linked list (right-skewed BST). This form of BST is known as backbone. The backbone form is reached by applying consecutive right rotations to all the nodes that has left child. Second step is consisting of consecutive left rotations to convert the BST to AVL. Calculate the number of nodes until second last level (n) and subtract it from the total number of nodes. This will give the number of nodes in last level. First rotate left as much as node at the last level. Then rotate left n/2, n/4, n/8 … 0 times. After consecutive rotations the BST is become an AVL tree which is also a complete binary tree.

Q3. Initial CustomSkipList has two level-list. Each item added the first level-list. The probability of the new item inserted to the next level is determined by the number of nodes between two tall neighbors. Let's call it n. So, the probability of the new item inserted to second level is % ((n / 10) * 100). The number of nodes between tall nodes can be found by counting the nodes between second level predecessor and successor. Second level is selecting, because second level contains the closest tall predecessor and successor compared to upper levels. After obtaining n, a random number is generated between 0 and 9. The probability of the generated random number smaller than n is %(10*n). With this random level selection, new nodes can be inserted as first or second level nodes.

A new level list is added to the skip list each time when the size of the first level list reaches multiples of 10. The level of all the tall nodes increased by one. This can be done by creating a new links array and linking the nodes to next node. Remove and Search operations are same as default skip-list. The search operation starts from the top-level list and ends at bottom level-list. During search operation an array of nodes are recorded for the predecessors of the target item. Both methods add and remove uses search to get the predecessors for creating new links or removing existing ones.

4. TEST CASES

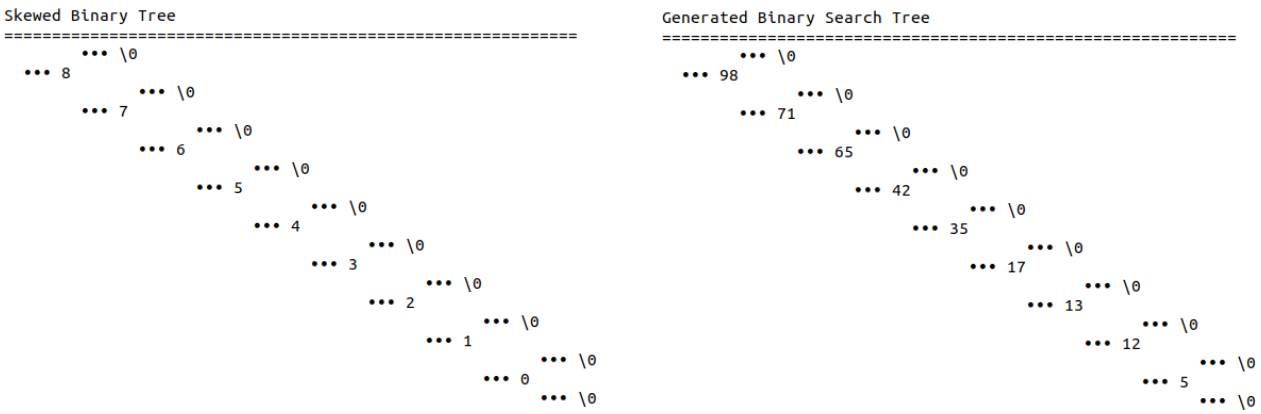
Q1						
Test ID	Test Case	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T1	Create skewed BST	1. Create a skewed binary tree which contains n item. 2. Create an array which has n unique item. 3. Call the method with created binary tree and array as parameters.	1. Skewed binary tree 2. array = {12, 42, 13, 65, 17, 35, 98, 5, 71};	Creates a skewed BST.	As Expected	**PASS**
T2	Create non-skewed BST	1. Create a non-skewed binary tree which contains n item. 2. Create an array which has n unique item. 3. Call the method with created binary tree and array as parameters.	1. Non-skewed binary tree 2. array = {12, 42, 13, 65, 17, 35, 98, 5, 71};	Creates a non-skewed BST.	As Expected	**PASS**
T3	Create a randomly structured BST	1. Create a random binary tree which contains n item. 2. Create an array which has n unique randomly generated item. 3. Call the method with created binary tree and array as parameters.	1. Randomly generated binary tree 2. Randomly generated array	Creates BST which has the same structure random binary tree.	As Expected	**PASS**

Q2						
Test ID	Test Case	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T1	Balance a skewed BST	1. Create a skewed BST. 2. Call the method with the BST.	Skewed BST	Converts the skewed BST to AVL tree.	As Expected	**PASS**
T2	Balance a non-skewed but unbalanced BST	1. Create a non-skewed BST. 2. Call the method with the BST.	Non-skewed but unbalanced BST	Converts the non-skewed but unbalanced BST to AVL tree.	As Expected	**PASS**
T3	Balance a randomly generated BST	1. Create a randomly generated BST. 2. Call the method with the BST.	Randomly generated BST	Converts the randomly generated BST to AVL tree.	As Expected	**PASS**

Q3						
Test ID	Test Case	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T1	Add	1. Create a data set that contains the items to insert skip-list. 2. Call the method add for each item inside the set.	Set = {35, 5, 20, 55, 15, 85, 90, 65, 45, 30, 95, 50, 10, 40, 25}	Adds 15 item into the skip-list and add a new level-list in 11. Insertion.	As Expected	**PASS**
T2	Remove	1. Create a data set that contains the items that going to be removed from skip-list. 2. Call the method remove for each item inside the set.	Set = {35, 5, 20, 55, 15, 85, 90, 65, 45, 30, 95, 50, 10, 40, 25}	Removes the given item from skip-list.	As Expected	**PASS**

T3	Find	1. Create a data set that contains the items being sought. 2. Call the method find for each item inside the set.	Set = { 12, 35, 20, 5, 17, 54, 15, 85, 65, 45, 30, 95, 10, 40, 25, 95, 71, 11, 0, 68, 69, 26}	Finds the given item.	As Expected	**PASS**
T4	Random	1. Create a randomly created data set that contains. 2. Add all the items. 3. Remove all the items.	Randomly created set.	Adds all the items in the set and removes all the items.	As Expected	**PASS**

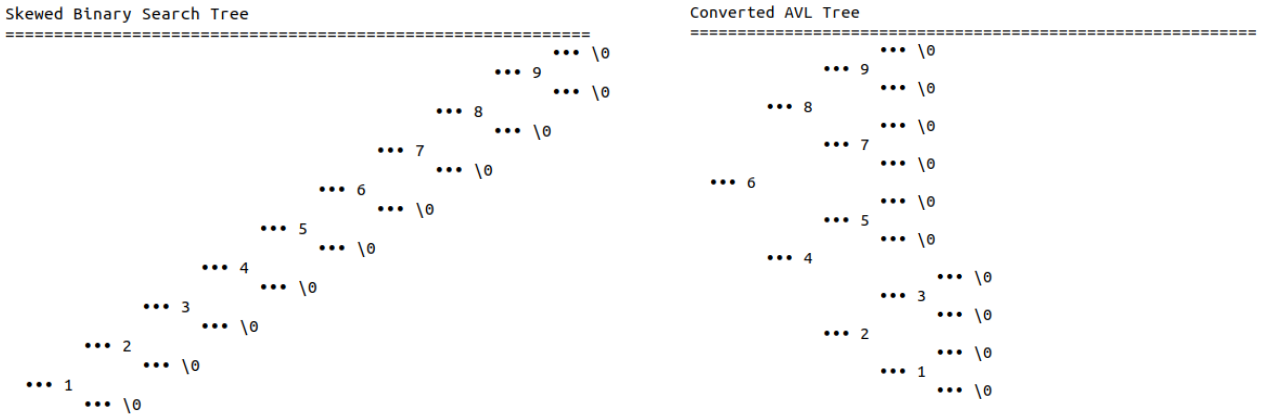
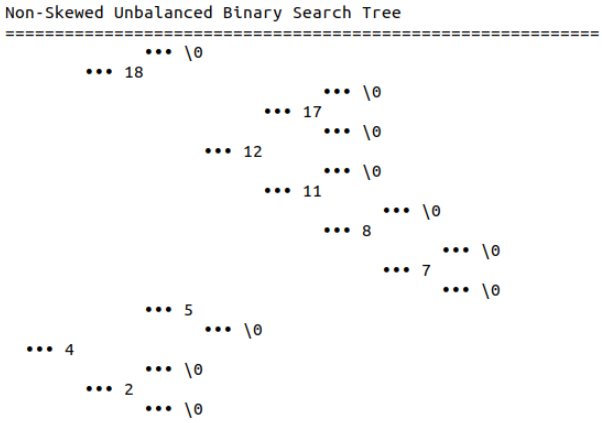
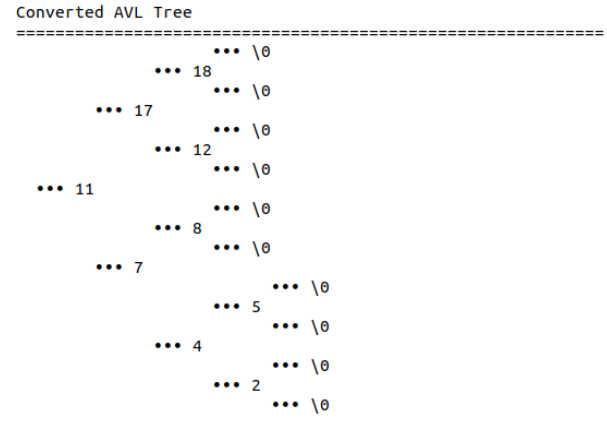
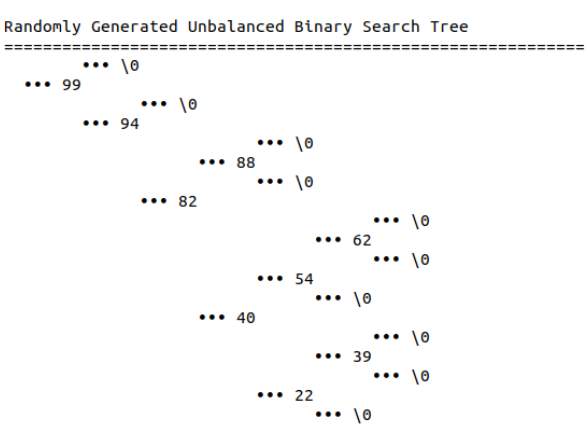
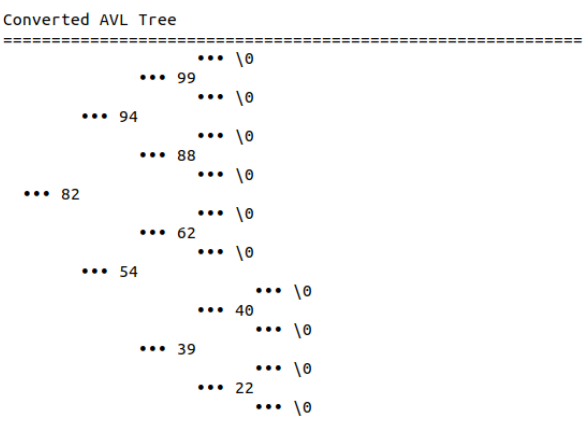
5. RUNNING AND RESULTS

Q1	
Test ID	Test Result
T1	

T2	Non-skewed Binary Tree ========== ••• \0 ••• 9 ••• \0 ••• 80 ••• \0 ••• 97 ••• \0 ••• 74 ••• \0 ••• 42 ••• \0 ••• 71 ••• \0 ••• 67 ••• \0 ••• 71 ••• \0 ••• 76 ••• \0	Generated Binary Search Tree ========== ••• \0 ••• 89 ••• \0 ••• 77 ••• \0 ••• 64 ••• \0 ••• 45 ••• \0 ••• 33 ••• \0 ••• 30 ••• \0 ••• 16 ••• \0 ••• 14 ••• \0 ••• 12 ••• \0
T3	Randomly Generated Binary Tree ========== ••• \0 ••• 55 ••• \0 ••• 83 ••• \0 ••• 40 ••• \0 ••• 41 ••• \0 ••• 89 ••• \0 ••• 60 ••• \0 ••• 99 ••• \0 ••• 85 ••• \0 ••• 78 ••• \0	Generated Binary Search Tree ========== ••• \0 ••• 97 ••• \0 ••• 92 ••• \0 ••• 60 ••• \0 ••• 56 ••• \0 ••• 38 ••• \0 ••• 22 ••• \0 ••• 21 ••• \0 ••• 20 ••• \0 ••• 6 ••• \0

- Note: The method have also been tested with larger input set (size = 1000) but they are not included in above table due to their difficulty in displaying clearly. You can check them in driver program.

Q2

Test ID	Test Result
T1	
T2	 
T3	 

- Note: The method have also been tested with larger input set (size = 1000) but they are not included in above table due to their difficulty in displaying clearly. You can check them in driver program.

Q3	
Test ID	Test Result
T1	1: 5 10 15 20 25 30 35 40 45 50 55 65 85 90 95 2: -- 10 -- -- 25 -- -- 40 -- 50 -- 65 85 3: -- -- -- -- -- -- -- -- -- -- -- 65 85
T2	1: 5 10 15 20 25 30 35 40 45 50 55 65 85 90 95 2: -- 10 -- -- 25 -- -- 40 -- 50 -- 65 85 3: -- -- -- -- -- -- -- -- -- -- -- 65 85 Remove: 35 1: 5 10 15 20 25 30 40 45 50 55 65 85 90 95 2: -- 10 -- -- 25 -- 40 -- 50 -- 65 85 3: -- -- -- -- -- -- -- -- -- -- 65 85 Remove: 5 1: 10 15 20 25 30 40 45 50 55 65 85 90 95 2: 10 -- -- 25 -- 40 -- 50 -- 65 85 3: -- -- -- -- -- -- -- -- -- 65 85 … 1: 10 15 25 30 40 45 50 55 65 85 90 95 2: 10 -- 25 -- 40 -- 50 -- 65 85 3: -- -- -- -- -- -- -- -- 65 85 Remove: 55 1: 10 15 25 30 40 45 50 65 85 90 95 2: 10 -- 25 -- 40 -- 50 65 85 3: -- -- -- -- -- -- -- 65 85 …

```
Remove: 15

1:  10   25   30   40   45   50   65   85   90   95
2:  10   25   --   40   --   50   65   85
3:  --   --   --   --   --   --   65   85


Remove: 85

1:  10   25   30   40   45   50   65   90   95
2:  10   25   --   40   --   50   65
3:  --   --   --   --   --   --   65
…

1:  10   25   40   50   95
2:  10   25   40   50
3:


Remove: 95

1:  10   25   40   50
2:  10   25   40   50
3:


…
```

```
1:  10   25   40
2:  10   25   40
3:


Remove: 10

1:  25   40
2:  25   40
3:


Remove: 40

1:  25
2:  25
3:


Remove: 25

1:
2:
3:
```

T3

```
Find 12        : false        Find 65        : true      Find 71        : false
Find 35        : true         Find 45        : true      Find 11        : false
Find 20        : true         Find 30        : true      Find 0         : false
Find 5         : true         Find 95        : true      Find 68        : false
Find 17        : false        Find 10        : true      Find 69        : false
Find 54        : false        Find 40        : true      Find 39        : false
Find 15        : true         Find 25        : true      Find 10        : true
Find 85        : true         Find 95        : true      Find 26        : false
```

T4

```
1:  0    14   15   50   67
2:


Add 55

1:  0    14   15   50   55   67
2:  --   --   --   --   55
```

… 11. item insertion (3rd level list added)

```
1:  0    13   14   15   50   51   55   67   77   89
2:  --   --   --   --   --   --   55   --   77
```

Add 30

```
1:  0    13   14   15   30   50   51   55   67   77   89
2:  --   --   --   --   30   --   --   55   --   77
3:  --   --   --   --   --   --   --   55   --   77
```

… 16. item insertion

```
1:  0    13   14   15   30   50   51   55   66   67   73   77   79   87   89
2:  --   --   --   --   30   --   --   55   --   --   --   77
3:  --   --   --   --   --   --   --   55   --   --   --   77
```

Add 62

```
1:  0    13   14   15   30   50   51   55   62   66   67   73   77   79   87   89
2:  --   --   --   --   30   --   --   55   62   --   --   --   77
3:  --   --   --   --   --   --   --   55   --   --   --   --   77
```

… 21. Item insertion (4th level list added)

Add 53

```
1:  0    13   14   15   16   30   30   50   51   53   55   62   66   67   73   75   77   79   87   89
2:  --   --   --   --   16   --   30   --   --   --   55   62   --   --   --   75   77
3:  --   --   --   --   --   --   --   --   --   --   55   --   --   --   --   --   77
```

Add 73

```
1:  0    13   14   15   16   30   30   50   51   53   55   62   66   67   73   73   75   77   79   87   89
2:  --   --   --   --   16   --   30   --   --   --   55   62   --   --   73   --   75   77
3:  --   --   --   --   --   --   --   --   --   --   55   62   --   --   --   --   75   77
4:  --   --   --   --   --   --   --   --   --   --   55   --   --   --   --   --   --   77
```

… 31. Item insertion (5th level list added)

```
1:  0   10   13   13   14   15   16   20   22   30   30   40   50   51   53   55   62   65   65   66   67   67   73   73   75   77   79   87   89   90
2:  --  10   --   --   --   --   16   20   --   --   30   --   --   --   --   55   62   --   65   --   --   --   73   --   75   77   --   --   --   90
3:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   62   --   --   --   --   --   --   --   75   77
4:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   --   --   --   --   --   --   --   --   --   77
```

Add 55

```
1:  0   10   13   13   14   15   16   20   22   30   30   40   50   51   53   55   55   62   65   65   66   67   67   73   73   75   77   79   87   89   90
2:  --  10   --   --   --   --   16   20   --   --   30   --   --   --   --   55   55   62   --   65   --   --   --   73   --   75   77   --   --   --   90
3:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   62   --   --   --   --   --   --   --   75   77
4:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   62   --   --   --   --   --   --   --   75   77
5:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   --   --   --   --   --   --   --   --   --   77
```

Add 72

```
1:  0   10   13   13   14   15   16   20   22   30   30   40   50   51   53   55   55   62   65   65   66   67   67   72   73   73   75   77   79   87   89   90
2:  --  10   --   --   --   --   16   20   --   --   30   --   --   --   --   55   55   62   --   65   --   --   --   --   73   --   75   77   --   --   --   90
3:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   62   --   --   --   --   --   --   --   --   75   77
4:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   62   --   --   --   --   --   --   --   --   75   77
5:  --  --   --   --   --   --   --   --   --   --   --   --   --   --   --   --   55   --   --   --   --   --   --   --   --   --   --   77
```

… Remove operations

```
1:  0   10  13  13  14  15  16  20  22  30  30  40  50  51  53  55  55  62  65  65  66  67  67  72  73  73  75  77  79  87  89  90
2:  --  10  --  --  --  --  16  20  --  --  30  --  --  --  --  55  55  62  --  65  --  --  --  --  73  --  75  77  --  --  --  90
3:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  --  75  77
4:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  --  75  77
5:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  --  --  --  --  --  --  --  --  --  --  77


Remove: 50

1:  0   10  13  13  14  15  16  20  22  30  30  40  51  53  55  55  62  65  65  66  67  67  72  73  73  75  77  79  87  89  90
2:  --  10  --  --  --  --  16  20  --  --  30  --  --  --  55  55  62  --  65  --  --  --  --  73  --  75  77  --  --  --  90
3:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  --  75  77
4:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  --  75  77
5:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  --  --  --  --  --  --  --  --  --  --  77


Remove: 0

1:  10  13  13  14  15  16  20  22  30  30  40  51  53  55  55  62  65  65  66  67  67  72  73  73  75  77  79  87  89  90
2:  10  --  --  --  --  16  20  --  --  30  --  --  --  55  55  62  --  65  --  --  --  --  73  --  75  77  --  --  --  90
3:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  --  75  77
4:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  --  75  77
5:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  --  --  --  --  --  --  --  --  --  --  77


Remove: 67

1:  10  13  13  14  15  16  20  22  30  30  40  51  53  55  55  62  65  65  66  67  72  73  73  75  77  79  87  89  90
2:  10  --  --  --  --  16  20  --  --  30  --  --  --  55  55  62  --  65  --  --  --  73  --  75  77  --  --  --  90
3:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  75  77
4:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  --  75  77
5:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  55  --  --  --  --  --  --  --  --  --  77
```

...

```
1:  10  13  16  20  22  30  40  53  55  62  65  65  66  67  72  73  75  90
2:  10  --  16  20  --  30  --  --  55  62  --  65  --  --  --  --  75  90
3:  --  --  --  --  --  --  --  --  55  62
4:  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  --  75
5:  --  --  --  --  --  --  --  --  55


Remove: 66

1:  10  13  16  20  22  30  40  53  55  62  65  65  67  72  73  75  90
2:  10  --  16  20  --  30  --  --  55  62  --  65  --  --  --  75  90
3:  --  --  --  --  --  --  --  --  55  62
4:  --  --  --  --  --  --  --  --  55  62  --  --  --  --  --  75
5:  --  --  --  --  --  --  --  --  55


Remove: 62

1:  10  13  16  20  22  30  40  53  55  65  65  67  72  73  75  90
2:  10  --  16  20  --  30  --  --  55  --  65  --  --  --  75  90
3:  --  --  --  --  --  --  --  --  55
4:  --  --  --  --  --  --  --  --  55  --  --  --  --  --  75
5:  --  --  --  --  --  --  --  --  55
```

... Last remove operation

```
1:  72
2:
3:
4:
5:


Remove: 72

1:
2:
3:
4:
5:
```