Problem 1

1. Algorithm reverse(str)

Input: A non-empty string separated by spaces, str of length n

Output: The input string in reversed order

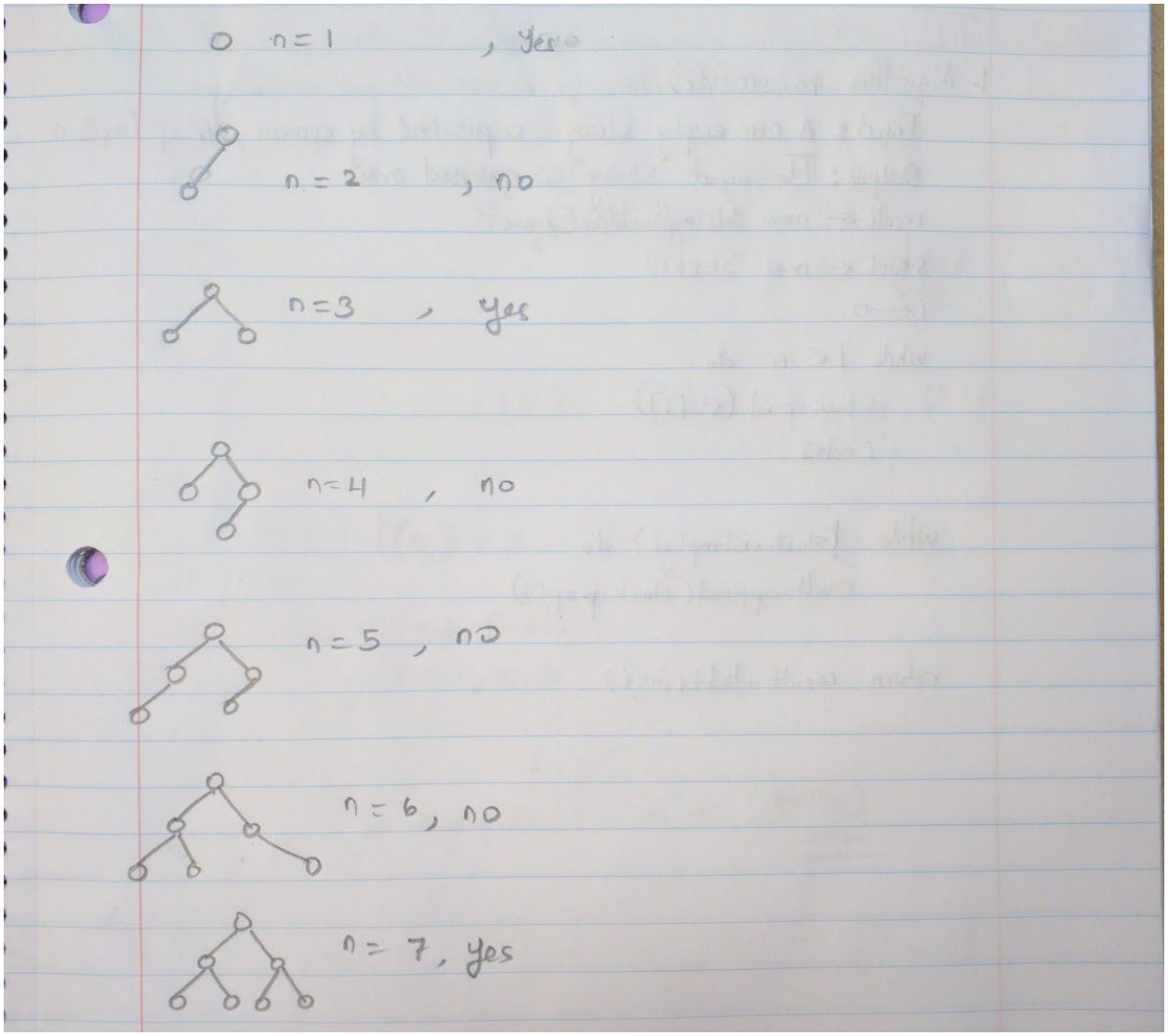
```
result ← new StringBuilder()
stack ← new Stack()
i ← 0
while i < n do
    stack.push(str[i])
    i = i+1

while !stack.isEmpty() do
    result.append(stack.pop())

return result.toString()
```

Problem 2

```
List<Integer> sortedArray = new LinkedList<>();
private void convertTreeToArray(Node t) {
   if (t != null) {
       printTree(t.left);
       sortedArray.add(t.element);
       printTree(t.right);
   }
}
@Override
public int[] sort(int[] arr) {
   int len = arr.length;
   int i = 0, j = 0;
   while (i < len){
       insert(arr[i++]);
   }
   convertTreeToArray(root);
   for(int elem : sortedArray){
       arr[j++] = elem;
   }
   return arr;
}
```

The asymptotic running time of the above algorithm is O(nlogn) because inserting n elements will take O(nlogn) and looping through the array and linkedList will be O(n) which makes it O(nlogn)

Problem 3



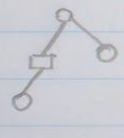
Problem 4

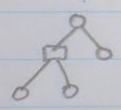
4.  1st node has to be black

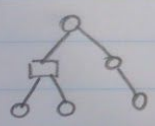
n = 1 , no

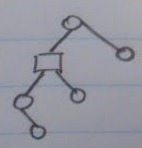
 n = 2 , yes.

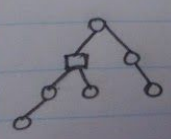
 n = 3 , no

 n = 4 , yes

 n = 5 , yes ,

 n = 6 , no

 n = 6 , no

 n = 7 , no.