Question 1

Implemented in python file

Question 2

a. *Implemented in python file*
b. *Implemented in python file*
c. It's impossible to create such a generator because there's a possibility of g being infinite and not containing any numbers divisible by 3. This means that we'll need to iterate over and infinite amount of numbers in search for the next one that's divisible by 3.
d. *Implemented in the python file*
e. It's impossible to create such a generator. In the case where one generator goes from -1 to -infinity and the other from 1 to -infinity we have to compare each item in the first generator with every item in the second since we cannot assume the sequences are well-ordered. So we'll need infinite time.
f. Impossible. in the case where the two generators are exactly the same, we'll need inifinte time to get to the next index where $a_i \neq b_i$.
g. *Implemented in python file*

Question 3

a. In the worst case scenario, the tree root has 2 nodes and each following node has a single child, n1 and n2 are leaves. In this case we'll need to visit every single node in the tree and perform O(1) work on each iteration making the complexity O(n).
b. *Skipped*
c. On each iteration the function splits the list in two and calls itself recursively for each of the halves. Meaning we visit each node and perform an insert operation costing O(d). Since $d = log(n + 1)$, the complexity of each iteration is O(log(n)). So the total time complexity is $O(n * log(n))$.
d. The function uses a binary search who's time complexity is $O(log(n))$ until k is found. And then the sum function goes over all the nodes below k for a total of **n** nodes while performing $O(1)$ work on each iteration. worst case scenario is k is the root time complexity of $O(n)$.

Question 4

a. *Implemented in python file*
b. The both the outer and the inner loop have a complexity of $O(n)$ for totalt of $O(n^2)$. on each iteration of the inner loop we perform 2 string splits, and a comparison of the cut strings costing O(k) each. So the total time complexity is $O(n^2 * k)$
c. Implemented in python file
d. **Dictionary creation**: the for loops costs $O(n)$. on each iteration the splitting of the string costs $O(k)$ and the insert costs $O(1)$. for a total of $O(n * k)$
Finding overlaps: outer loop costs $O(n)$. on each iteration splitting the string and finding it in the dictionary costs $O(k)$. on average there'll be 1 entry for each key in the dictionary

so the inner loops will have one iteration $O(1)$ then the append function to the list also costs $O(1)$. in total the complexity is $n * O(k) + n * O(k) = O(n * k)$.

e. *Implemented in python file*

Question 5

a. *נניח בשלילה כי קיים מספר רציונלי שלא ניתן לייצג כמנה של מספרים שלמים וזרים. יהיו p,q מספרים שלמים וזרים. מכך שהם זרים קיים n=gcd(p,q) ששונה מ-1. משמע, ניתן לחלק את המונה והמכנה. כלומר $(p/n)/(q/n)$. ומתקבל שבר שקול לשבר המקורי בסתירה להנחה. מכאן נובע כי כל מספר רציונלי ניתן לייצג כמנה של מספרים שלמים וזרים.*

b. *Implemented in python file*

c. *Implemented in python file*