**Problem 1 (Fibonacci Sayıları)**

Fibonacci dizisindeki her yeni terim, önceki iki terimeklenerek oluşturulur. 1 ve 2 ile başlayarak, ilk 10 terim şöyle olacaktır:

1. liste = [1,2] # başlangıç terimlerinibelirledik.
2. toplam = 0 # sonuç da göstereceyiğimiztoplamıtanımladık.
3. for i in range(2,100): # ilk 100 terimebaktım. Bu aslındaiyibirçözümsayılmaz
4. fib = liste[i-2]+liste[i-1] # seçilenterimveöncekiterimitopladım.
5. if fib > 4000000: #4 milyanuaşmıssakeğerdöngüyükırıyoruz.
6. break
7. liste.append(fib) # fibonaccisayılarımızlisteyeekliyoruz.
8. for i in liste:
9. if i%2 ==0 :
10. toplam= toplam+i # toplamıbul
11. print(liste) # tümsayılarıgöster.
12. print(toplam) #Toplamıgöster.

**Problem 11 (Son derece bölünebilir üçgen sayı)**

1. [codelang=”fsharp”]
2. openSystem
3. lettriangleNumber(n:int64) = [1L..n] |>Seq.sum
4. letfindFactorsOf(n:int64) =
5. letupperBound = int64(Math.Sqrt(double(n)))
6. [1L..upperBound]
7. |>Seq.filter (funx -> n % x = 0L)
8. |>Seq.collect (funx -> [x; n/x])
9. letnaturalNumbers = Seq.unfold (fun x ->Some(x, x+1L)) 1L
10. letanswer =naturalNumbers
11. |>Seq.map (funx ->triangleNumber(x))
12. |>Seq.filter (funx ->Seq.length(findFactorsOf(x)) >= 500)
13. |>Seq.head
14. [/code]

**Problem 21 (İsim Sorları)**

1. functionnamesScores(arr) {
2. arr.unshift(0)
3. arr = arr.sort()
4. functionnameScoreCalc(word, i){
5. letsum = 0
6. letscore;
7. letalpha = [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]
8. for (let i = 0 ; i <word.length; i++){
9. sum+= alpha.indexOf(word[i].toLowerCase())
10. }
11. returnsum\*i
12. }
13. let total = 0
14. for (let i = 0 ; i <arr.length; i++){
15. total +=nameScoreCalc(arr[i], i)
16. }
17. return total
18. }
19. const test1 = ['THIS', 'IS', 'ONLY', 'A', 'TEST'];
20. const test2 = ['I', 'REPEAT', 'THIS', 'IS', 'ONLY', 'A', 'TEST'];

22.namesScores(test1);

**Problem 191 (Ödül Dizesi)**

1. varsayi = 0;
2. var ara = function(day,consecA,lateOnce) {
3. varsayi = 0;
4. if(day == 30) {return 1;}
5. if(consecA< 2) {
6. sayi += ara(day+1,consecA+1,lateOnce);
7. }
8. İf(!lateOnce) {
9. sayi += search(day+1,0,true);
10. }
11. sayi += ara(day+1,0,lateOnce);
12. returnsayi;
13. }
14. console.log(ara(0,0,false));

**Problem 1 (3 ve 5’in Katları)**

1. functionsayi(limit) {
2. var i, toplam =0;
3. for (let i = 3; i< limit; i++) {
4. if (i % 3 === 0 || i % 5 === 0) {
5. toplam += 1;
6. }
7. }
8. return toplam;
9. }
10. Console.log (“Cevap = ” + sayi(1000));

**LEETCODE**

**PROBLEM 1 (İki Numara Ekle)**

1. public ListNodeaddTwoNumbers(ListNode l1, ListNode l2) {
2. ListNodedummyHead = new ListNode(0);
3. ListNode p = l1, q = l2, curr = dummyHead;
4. int carry = 0;
5. while (p != null || q != null) {
6. int x = (p != null) ? p.val : 0;
7. int y = (q != null) ? q.val : 0;
8. int sum = carry + x + y;
9. carry = sum / 10;
10. curr.next = new ListNode(sum % 10);
11. curr = curr.next;
12. if (p != null) p = p.next;
13. if (q != null) q = q.next;
14. }
15. if (carry > 0) {
16. curr.next = new ListNode(carry);
17. }
18. return dummyHead.next;
19. }

**PROBLEM 21(Parantez Oluştur)**

1. class Solution {
2. publicList<String>generateParenthesis(int n) {
3. List<String>combinations = newArrayList();
4. generateAll(newchar[2 \* n], 0, combinations);
5. returncombinations;
6. }
7. publicvoidgenerateAll(char[] current, int pos, List<String>result) {
8. if (pos == current.length) {
9. if (valid(current))
10. result.add(newString(current));
11. } else {
12. current[pos] = '(';
13. generateAll(current, pos+1, result);
14. current[pos] = ')';
15. generateAll(current, pos+1, result);
16. }
17. }
18. publicbooleanvalid(char[] current) {
19. intbalance = 0;
20. for (char c: current) {
21. if (c == '(') balance++;
22. elsebalance--;
23. if (balance< 0) returnfalse;
24. }
25. return (balance == 0);
26. }
27. }
28. defgenerateParenthesis(self, N):
29. if N == 0: return ['']
30. ans = []
31. for c in xrange(N):
32. forleft in self.generateParenthesis(c):
33. forright in self.generateParenthesis(N-1-c):
34. ans.append('({}){}'.format(left, right))
35. returnans

**PROBLEM 41 (Yağmur Suyunun Yakalanması)**

1. int trap(vector<int>&height)
2. {
3. if(height == null)
4. return 0;
5. intans = 0;
6. int size = height.size();
7. vector<int>left\_max(size), right\_max(size);
8. left\_max[0] = height[0];
9. for (int i = 1; i < size; i++) {
10. left\_max[i] = max(height[i], left\_max[i - 1]);
11. }
12. right\_max[size - 1] = height[size - 1];
13. for (int i = size - 2; i >= 0; i--) {
14. right\_max[i] = max(height[i], right\_max[i + 1]);
15. }
16. for (int i = 1; i < size - 1; i++) {
17. ans += min(left\_max[i], right\_max[i]) - height[i];
18. }
19. returnans;
20. }

**PROBLEM 771 (Mücevherler ve Taşlar)**

1. varnumJewelsInStones = function(J, S) {
2. letsayi = 0
3. for(i =0; i<S.length; i++) {
4. for(k =0; k<J.length; k++) {
5. İf (S.charAt(i) === J.charAt(k)) {
6. sayi ++;
7. }
8. }
9. }
10. returnsayi
11. };

**PROBLEM 191 (1 Bit Sayısı)**

1. varhammingWeight = function(n) {
2. returnn.ToString(2).replace(/0/g, ‘’).length