**19I510 Design and Analysis of Algorithms**

**Exercise 1 – Analysis of Algorithms**

**Asymptotic Analysis of Algorithms:**

The objective of asymptotic analysis is to have a measure of efficiency of algorithms that doesn’t depend on machine specific constants, and doesn’t require algorithms to be implemented and time taken by programs to be compared. Asymptotic notations are mathematical tools to represent time complexity of algorithms for asymptotic analysis. Best case denoted by Ω Notation, Worst case denoted by Big O Notation and Average case denoted by Θ Notation.

1. Find the last digit of nth Fibonacci number. Hint: Fibonacci numbers can grow exponentially fast.

= 280 571 172 992 510 140 037 611 932 413 038 677 189 525 if the naïve solution [compute Fibonacci and determine the last digit] is used, it will fail . Find effective way to address the issue.

**Input Format**Input consists of single integer n   
**Output Format**Output the last digit of

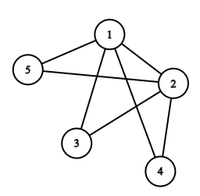
**Sample Input**

3

**Sample Output**  
2

1. Given two positive integers **N** and **K**, the task is to construct a simple and connected graph consisting of **N** vertices with the length of each edge as **1** unit, such that the shortest distance between exactly **K** pairs of vertices is **2**. If it is not possible to construct the graph, then print **-1**. Otherwise, print the edges of the graph.

**Input Format**Input consists of single integer N, K such that N<30 and K<N\*Ns  
**Output Format**Output pair of vertices as {x,y}



**Sample Input:** N = 5, K = 3

**Sample Output:** (3, 4), (4, 5), (3, 5)

**Sample Input:** N = 5, K = 8

**Sample Output:** -1

1. Given a Binary Search Tree (BST) with N nodes and a positive integer k, find the k’th largest element in the Binary Search Tree.   
   For example, in the following BST, if k = 3, then output should be 14, and if k = 5, then output should be 10.



**Input Format**Input consists of single integer N, K such that N<30 and k<N  
**Output Format**Output the kth value

**Sample Input:** k = 3

**Sample Output:** 14

**Sample Input:** k=5

**Sample Output:** 10

1. Compute the GCD of two numbers using Euclid’s algorithm.

It's based on the idea that if you have two numbers, let's say a and b. Assume a > b without loss of generality.

1. The gcd of a and b will be the same as that of b and a - b.

It's possible that a >> b (much larger), in such cases subtracting b one by one, is pointless, slow, and not required as we can find the remainder when we divide a with b, i.e., gcd(a, b) = gcd(a % b, b), this is equivalent to subtracting b as long as the result is not a negative number.

1. Now, after this step (a % b) < b, we continue the process of finding the remainder as the above for b and a % b until we end up with b=0. The other leftover number, in that case, will be the gcd of a and b.

Example.

Let, a = 33, b = 15

=> gcd(a, b)

=> gcd(a % b, b)

=> gcd(3, 15)

=> gcd(3, 15 % 3)

=> gcd(3, 0)

**Input Format**Input consists of two integers a and b in the range 1 to 10^7

**Output Format**Output the greatest common divisor

**Sample Input**

15 33

**Sample Output**  
3