import java.util.Scanner;

public class FibonacciRecursive {

private static int stepCount;

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of Fibonacci terms: ");

int n = scanner.nextInt();

System.out.print("Fibonacci Series: ");

for (int i = 0; i < n; i++) {

stepCount = 0; // Reset for each term

System.out.print(fibonacci(i) + " ");

}

System.out.println();

System.out.print("Enter position (0 to " + (n - 1) + "): ");

int pos = scanner.nextInt();

stepCount = 0; // Reset for specific position

System.out.printf("Fibonacci(%d) = %d (Steps: %d)%n", pos, fibonacci(pos), stepCount);

scanner.close();

}

public static int fibonacci(int n) {

stepCount++;

return (n <= 1) ? n : fibonacci(n - 1) + fibonacci(n - 2);

}

}

#iterative

import java.util.Scanner;

public class FibonacciIterative {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of Fibonacci terms: ");

int n = scanner.nextInt();

System.out.print("Fibonacci Series: ");

for (int i = 0; i < n; i++) {

System.out.print(fibonacci(i) + " ");

}

System.out.println();

System.out.print("Enter position (0 to " + (n - 1) + "): ");

int pos = scanner.nextInt();

System.out.printf("Fibonacci(%d) = %d (Steps: %d)%n", pos, fibonacci(pos), n - 1);

scanner.close();

}

public static int fibonacci(int n) {

if (n == 0) return 0;

if (n == 1) return 1;

int a = 0, b = 1;

for (int i = 2; i <= n; i++) {

int c = a + b;

a = b;

b = c;

}

return b;

}

}

#job\_sequencing

import java.util.Scanner;

class Job {

int id, deadline, profit;

Job(int id, int deadline, int profit) {

this.id = id; this.deadline = deadline; this.profit = profit;

}

}

public class JobSequencing {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of jobs: ");

int n = scanner.nextInt();

Job[] jobs = new Job[n];

for (int i = 0; i < n; i++) {

System.out.print("Enter job ID, deadline, profit: ");

jobs[i] = new Job(scanner.nextInt(), scanner.nextInt(), scanner.nextInt());

}

scheduleJobs(jobs, n);

scanner.close();

}

public static void scheduleJobs(Job[] jobs, int n) {

for (int i = 0; i < n - 1; i++) {

int maxIndex = i;

for (int j = i + 1; j < n; j++) {

if (jobs[j].profit > jobs[maxIndex].profit) maxIndex = j;

}

Job temp = jobs[i]; jobs[i] = jobs[maxIndex]; jobs[maxIndex] = temp;

}

int[] scheduledJobs = new int[n]; int totalProfit = 0;

for (int i = 0; i < n; i++) {

int deadline = jobs[i].deadline;

if (deadline > n) deadline = n;

for (int j = deadline - 1; j >= 0; j--) {

if (scheduledJobs[j] == 0) {

scheduledJobs[j] = jobs[i].id; totalProfit += jobs[i].profit; break;

}

}

}

System.out.print("Scheduled jobs: ");

for (int id : scheduledJobs) if (id != 0) System.out.print(id + " ");

System.out.println("\nTotal Profit: " + totalProfit);

}

}

#fractional

import java.util.Arrays;

import java.util.Scanner;

public class FractionalKnapSack {

static class Item {

int profit, weight;

Item(int profit, int weight) { this.profit = profit; this.weight = weight; }

}

private static double getMaxValue(Item[] items, int capacity) {

Arrays.sort(items, (a, b) -> Double.compare((double)b.profit / b.weight, (double)a.profit / a.weight));

double totalValue = 0;

for (Item item : items) {

if (capacity == 0) break;

int weightTaken = Math.min(item.weight, capacity);

double fraction = (double) weightTaken / item.weight;

double profitGained = item.profit \* fraction;

int gcd = gcd(weightTaken, item.weight);

System.out.printf("Item (profit: %d, weight: %d) - Taken: %.2f%%, Fraction: %d/%d, Profit gained: %.2f\n",

item.profit, item.weight, fraction \* 100, weightTaken / gcd, item.weight / gcd, profitGained);

totalValue += profitGained;

capacity -= weightTaken;

}

return totalValue;

}

private static int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of items: ");

int n = scanner.nextInt();

Item[] items = new Item[n];

for (int i = 0; i < n; i++) {

System.out.print("Enter profit and weight for item " + (i + 1) + ": ");

items[i] = new Item(scanner.nextInt(), scanner.nextInt());

}

System.out.print("Enter capacity: ");

int capacity = scanner.nextInt();

double maxValue = getMaxValue(items, capacity);

System.out.printf("Maximum value in Knapsack = %.2f\n", maxValue);

scanner.close();

}

}

#0/1

import java.util.Scanner;

public class Knapsack {

public static int knapsack(int[] profits, int[] weights, int capacity) {

int n = profits.length;

int[][] dp = new int[n + 1][capacity + 1];

// Fill the DP table

for (int i = 1; i <= n; i++)

for (int w = 1; w <= capacity; w++)

dp[i][w] = (weights[i - 1] <= w)

? Math.max(dp[i - 1][w], profits[i - 1] + dp[i - 1][w - weights[i - 1]])

: dp[i - 1][w];

System.out.println("Max Profit: " + dp[n][capacity]);

System.out.print("Items: ");

// Find the items to include in the knapsack

for (int i = n, w = capacity; i > 0 && dp[i][w] > 0; i--) {

if (dp[i][w] != dp[i - 1][w]) {

System.out.print(i + " "); // Print item index (1-based)

w -= weights[i - 1]; // Update remaining weight

}

}

return dp[n][capacity];

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input number of items

System.out.print("Enter the number of items: ");

int n = scanner.nextInt();

int[] profits = new int[n];

int[] weights = new int[n];

// Input profits and weights for each item

for (int i = 0; i < n; i++) {

System.out.print("Enter profit and weight for item " + (i + 1) + ": ");

profits[i] = scanner.nextInt();

weights[i] = scanner.nextInt();

}

// Input capacity of the knapsack

System.out.print("Enter the capacity of the knapsack: ");

int capacity = scanner.nextInt();

// Call the knapsack function

knapsack(profits, weights, capacity);

scanner.close();

}

}