Задача 1.

Вводная

**Неправильные числа Фибоначчи**

**Ограничение по времени работы программы: 1 секунда**

Зададим «неправильные» числа Фибоначчи, в которых n-е число равно сумме n−1-го и n−3-го. Более формально,

F0=F1=F2=1,

Fn=Fn−1+Fn−3, при n>2.

Вычислите n-е «неправильное» число Фибоначчи  Попробуйте реализовать рекурсивную функцию, даже если вы знаете другие (более эффективные) способы вычисления.

ВХОДНЫЕ ДАННЫЕ

Программа получает на вход одно целое число n, 0⩽n⩽35.

ВЫХОДНЫЕ ДАННЫЕ

Программа должна вывести одно искомое число.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 7 | 9 |

Начало формы

Задача 2.

Лёгкая

**Функция Аккермана**

**Ограничение по времени работы программы: 1 секунда**

В теории вычислимости важную роль играет функция Аккермана A(m,n) от двух параметров (неотрицательных целых чисел), определенная следующим образом:

A(m,n)=n+1, при m=0,

A(m,n)=A(m−1,1), при m>0, n=0,

A(m,n)=A(m−1,A(m,n−1)), при m>0, n>0.

По данным m и n вычислите A(m,n).

ВХОДНЫЕ ДАННЫЕ

Программа получает на вход два целых неотрицательных числа m и n, записанных в отдельных строчках.

ВЫХОДНЫЕ ДАННЫЕ

Программа должна вывести значение A(m,n). Известно, что результат не превосходит 100000.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 32 | 29 |

Начало формы

Задача 3.

Средняя

**Быстрое возведение в степень**

**Ограничение по времени работы программы: 1 секунда**

Даны два числа - a и n. Вычислите три последние цифры числа an.

ВХОДНЫЕ ДАННЫЕ

В первой строке входных данных записано число a, во второй строке - число n. Оба числа - натуральные, не превосходящие 109.

ВЫХОДНЫЕ ДАННЫЕ

Программа должна вывести три последние цифры числа an. При этом лидирующие нули в ответе можно не выводить.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 216 | 536 |
| 210 | 24 |

Начало формы

Задача 4.

Средняя

**Ханойские башни - количество операций**

**Ограничение по времени работы программы: 1 секунда**

Дано число n. Какое наименьшее число перекладываний дисков в головоломке «Ханойские башни» нужно совершить для переноса пирамидки из n дисков c одного стержня на другой?

ВХОДНЫЕ ДАННЫЕ

Программа получает на вход одно натуральное число n, 1⩽n⩽15.

ВЫХОДНЫЕ ДАННЫЕ

Программа должна вывести одно натуральное число - минимальное число перекладываний дисков, необходимых для переноса пирамидки из n дисков с одного стержня на другой.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 2 | 3 |

Начало формы

Решение задачи №1

Необходимо аккуратно запрограммировать рекуррентные соотношения.

РЕШЕНИЕ НА ЯЗЫКЕ PYTHON

def fib(n):
    if n <= 2:
        return 1
    else:
        return fib(n - 1) + fib(n - 3)

print(fib(int(input())))

Решение задачи №2

Необходимо аккуратно запрограммировать условия задачи.

РЕШЕНИЕ НА ЯЗЫКЕ PYTHON

При решении на языке Python придется увеличить максимальную глубину рекурсии при помощи sys.setrecursionlimit.

import sys
sys.setrecursionlimit(5000)
def A(m, n):
    if m == 0:
        return n + 1
    if m > 0 and n == 0:
        return A(m - 1, 1)
    if m > 0 and n > 0:
        return A(m - 1, A(m, n - 1))

m = int(input())
n = int(input())
print(A(m, n))

РЕШЕНИЕ НА ЯЗЫКЕ СИ

#include <stdio.h>
int A(int m, int n)
{
    if (m == 0)
        return n + 1;
    if (m > 0 && n == 0)
        return A(m - 1, 1);
    if (m > 0 && n > 0)
        return A(m - 1, A(m, n - 1));
}

int main()
{
    int m, n;
    scanf("%d%d", &n, &m);
    printf("%d\n", A(m, n));
    return 0;
}

Решение задачи №3

Необходимо модифицировать алгоритм быстрого возведения в степень так, чтобы после каждой операции брался остаток от деления результата на 1000. Затем вычислим остаток от деления на 1000  an при помощи модифицированного алгоритма.

РЕШЕНИЕ НА ЯЗЫКЕ PYTHON

def power(a, n):
    if n == 0:
        return 1
    elif n % 2 == 1:
        return power(a, n - 1) \* a % 1000
    else:
        return power(a, n // 2) \*\* 2 % 1000

a = int(input())
n = int(input())
print(power(a, n))

РЕШЕНИЕ НА ЯЗЫКЕ C

#include<stdio.h>

int power(int a, int n)
{
    a %= 1000;
    if (n == 0)
        return 1;
    else if (n % 2 == 1)
        return a \* power(a, n - 1) % 1000;
    else
        return power(a \* a % 1000, n / 2);
}

int main()
{
    int a, n;
    scanf("%d%d", &a, &n);
    printf("%d\n", power(a % 1000, n));
}

Решение задачи №4

Модифицируем алгоритм "Ханойских башен" так, чтобы вместо вывода последовательности перемещений на экран он просто подсчитывал их количество. Первый способ сделать это — завести глобальную переменную ans и вместо печати перемещения увеличивать значение ans на 1:

РЕШЕНИЕ НА ЯЗЫКЕ PYTHON

ans = 0

def move(n, start, finish):
    global ans
    if n > 0:
        temp = 6 - start - finish # Вспомогательный стержень
        move(n - 1, start, temp)
        ans += 1
        move(n - 1, temp, finish)

n = int(input())
move(n, 1, 2)
print(ans)

Конец формы

Конец формы

Конец формы

Конец формы