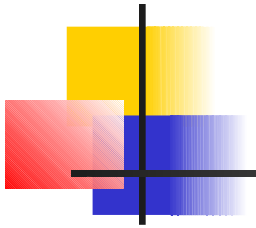
Объектно - ориентированное программирование систем защиты информации

Васина А.В.
Сидоров А.С.
БПЗ 1101

Задача A. Маленькие кубики

- Имя входного файла: *input.txt*
- Имя выходного файла: *output.txt*
- Ограничение по времени: *1 second*
- Ограничение по памяти: *64 megabytes*

Вася любит пространственные фигуры из кубиков. Все его кубики одинакового размера. Недавно он построил новую пространственную фигуру и теперь хочет посчитать, сколько кубиков лежат строго внутри этой фигуры.



- **Формат входного файла**

В первой строке записаны 3 числа n, m, k $(1 \leq n, m, k \leq 50)$ — длина, ширина и высота фигуры.
Далее с новой строки записаны k блоков, каждый из которых состоит из n строк по m символов “0” или “1”. “1” показывает, что кубик на этом месте есть, а “0” — что его нет.

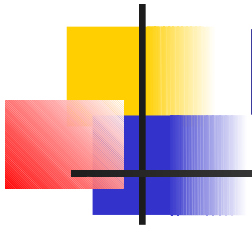
- **Формат выходного файла**

Выведите единственное число - число кубиков, расположенных

строго внутри фигуры

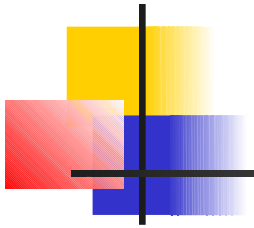
- **Примеры**

input.txt	output.txt
3 3 3 111 111 111 111 111 111 111 111 111	1



Код программы

- import java.io.File;
- import java.io.IOException;
- import java.io.PrintWriter;
- import java.util.ArrayList;
- import java.util.Arrays;
- import java.util.Scanner;



- public class TaskA{

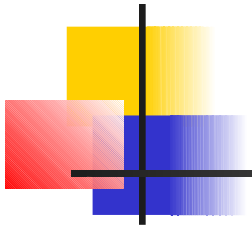
public static void main(String[] args)throws IOException{

- Scanner in = new Scanner(new File("input.txt"));

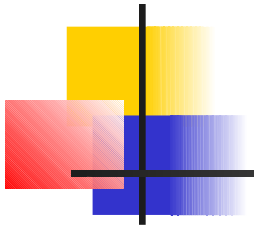
// считывание из файла

- PrintWriter out =new PrintWriter(newFile("output.txt"));

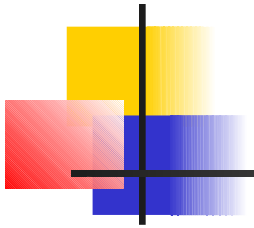
//записывание в файл



- *n = in.nextInt();*
- *m = in.nextInt();*
- *k = in.nextInt();*



```
a = new int[k][n][m];
in.nextLine();
for(int i=0; i<k; i++){
    for(int j=0; j<n; j++){
            String s = in.nextLine();
                for(int l=0; l<m; l++)
                    a[i][j][l] = s.charAt(l)-48;
                }
            }
```



```
int ans=0;
for(int i=0; i<k; i++)
     for(int j=0; j<n; j++)
          for(int l=0; l<m; l++)
               if(check(i,j,l)) ans++;
out.print(ans);
out.close();

}
```

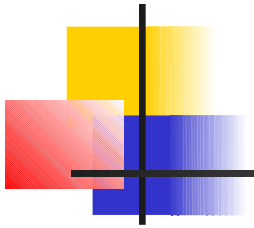
```
static int n,m,k;
static int[][][] a;
```

```
    static boolean check(int l,int i, int j){
        if(l>0 && l<k-1 && i>0 && i<n-1 && j>0 &&
  j<m-1){
            return a[l][i][j]==1 &&
                a[l-1][i][j]==1 && a[l+1][i][j]==1 &&
                a[l][i+1][j]==1 && a[l][i-1][j]==1 &&
                a[l][i][j+1]==1 && a[l][i][j-1]==1;


        }else return false;
    }

}
```

Задача B. Отличные числа

- Имя входного файла: *input.txt*
- Имя выходного файла: *output.txt*
- Ограничение но времени: *1 second*
- Ограничение по памяти: *64 megabytes*

- Отличница Катя очень любит получать пятерки в школе и пятерки вообще. По мнению Кати, число называется отличным, если оно имеет ровно 5 делителей. Помогите Кате посчитать, сколько существует отличных чисел на, отрезке [1, n].



- Формат входного файла

В единственной строке записано одно число n (1 ≤ n ≤ 10^18)

- Формат выходного файла

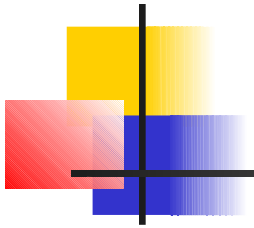
Выведите одно число – количество чисел на отрезке [1,n]

- Пример

1	0
1000	3
1000000	11

Код программы

- import java.util.Scanner;
- import java.io.File;
- import java.io.IOException;
- import java.io.PrintWriter;
- import java.math.*;



- public class TaskB{

- public static void main(String[] args)throws IOException{

Scanner in = new Scanner(new File("input1.txt")); *//считываем из файла*

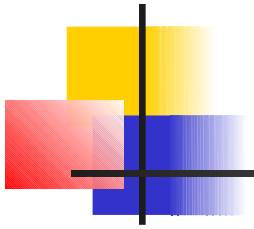
PrintWriter out =newPrintWriter(newFile("output2.txt")); *//записываем в файл*

- long n = in.nextLong();

//берем переменную из файла

- int ans = 0;

```
boolean flag = true; //переменная имеющая значения true или false
        for(int i=2; flag; i++){
                if(isPrime(i)){
                        long y = 1;
                        for(int j=0; j<4; j++)
                                y*=(long)i;
                        System.out.print(y);
                if(y>n || y<0) flag = false; else ans++;

                }
        }
out.print(ans);//записываем ответ в блокнот
out.close();//закрываем его

}
```

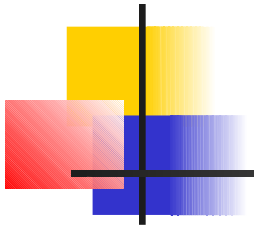


```
static boolean isPrime(int x){
 //выводит true если число целое, в другом случае
    false
        for(int i=2; i<=Math.sqrt(x); i++)
            if(x%i==0) return false;
        return true;
}
```

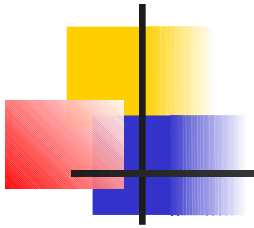
Задача C. Числа - палиндромы

- Имя входного файла: *input.txt*
- Имя выходного файла: *output.txt*
- Ограничение по времени: *1 second*
- Ограничение по памяти: *64 megabytes*

- Девочка, Анна, очень любит палиндромы. Недавно она, захотела, посчитать, сколько существует положительных чисел - палиндромов без лидирующих нулей длины не более л, но не смогла. Помогите ей.



- Палиндром - число (например, 404), буквосочетание, слово или текст, одинаково (или почти одинаково) читающиеся в обоих направлениях.



- Формат входного файла

В единственной строке записано единственное число n (1 < *n* < 50).

- Формат выходного файла

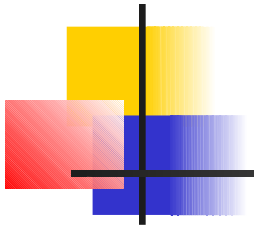
Выведите единственное число: количество положительных чисел - палиндромов без лидирующих нулей длины не более *n*
по модулю 10^9 + 7.

- Примеры

1	9
2	18
3	108

Код программы

- import java.io.*;
- import java.util.*;//подключение библиотеки для использования сканера



- public class TaskC{

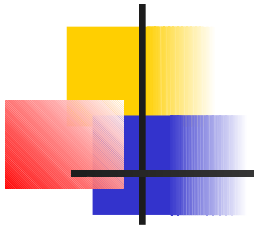
public static void main(String[] args)throws IOException{

- Scanner in = new Scanner(new File("input.txt"));

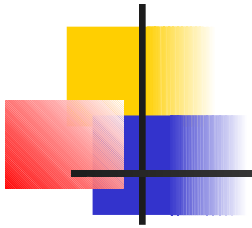
// считывание из файла

- PrintWriter out =new PrintWriter(newFile("output.txt"));

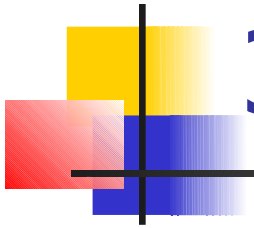
//записывание в файл



```
int n = in.nextInt(); //берем значение из блокнота
long ans=0; // в ответе изначально 0

        for(int i=1; i<=n; i++){
                long a = 9;
                for(int j=0; j<(i+1)/2-1; j++)
                        a = (a*10)%1000000007;
                ans = (ans+a)%1000000007;

        }
```



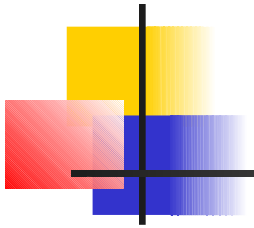
```
out.print(ans);
//записываем результат в блокнот
out.close();
//Закрываем его
   }
}
```



Задача D. Пивной вор

- Имя входного файла: *input.txt*
- Имя выходного файла: *output.txt*
- Ограничение но времени: *1 second*
- Ограничение по памяти: *64 megabytes*

- Однажды алкоголик Борис пробрался в магазин и увидел там n ящиков пива различной стоимости. Какую сумму может сэкономить алкоголик Борис, если он может унести с собой не более чем k ящиков пива?



Формат входного файла

В первой строке содержатся два числа n и k ($1 \le n, k \le 100000$) –

количество ящиков пива в магазине и максимальное число

ящиков, которое может унести с собой алкоголик Борис.

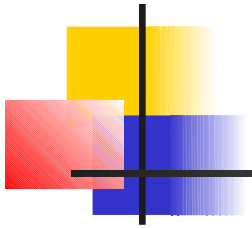
Во второй строке содержится n чисел $a_1 \ldots a_n$ ($1 \le a_i \le 10^9$) –

стоимости ящиков пива, находящихся в магазине.

Формат выходного файла

Выведите единственное число – максимальная сумма, которую

может сэкономить алкоголик Борис, украв из магазина некоторое



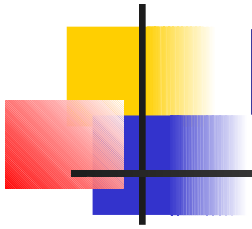
- Пример

6 4 34

7 3 10 8 1 9

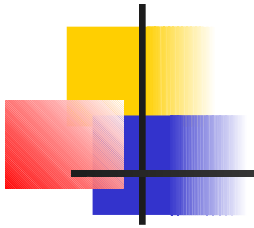
3 8 18

5 4 9

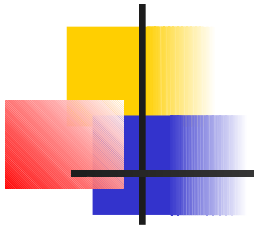


Код программы

- **import** java.io.File;
- **import** java.io.IOException;
- **import** java.io.PrintWriter;
- **import** java.util.Arrays;
- **import** java.util.Scanner;



- **public class** LabD{
- **public static void** main(String[] args)**throws** IOException{
- Scanner in = **new** Scanner(**new** File("input.txt"));
- PrintWriter out =**new** PrintWriter(**new** File("output.txt"));



- **int** n = in.nextInt();

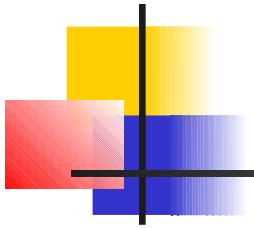
// присваиваем n кол-во ящиков в магазине

- **int** k = in.nextInt();

// присваиваем k кол-во ящиков которые можем унести

- **int**[] a = **new int**[n];

// создаем массив a размерностью n



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

// записываем в массив значения

- a[i] = in.nextInt();
- Arrays.*sort*(a);

//сортирует массив по возрастанию

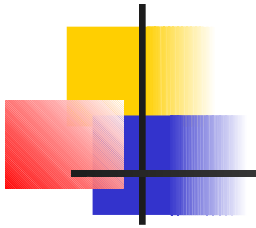
- **long** ans = 0;
- **for**(**int** i=n-1,j=0; j<Math.*min*(n, k); i--,j++)

// начинаем смотреть значения с конца
// если i=n-1 то берется последнее значение, после чего j(кол-во взятых ящиков) увеличивается на единицу, пока не перестанет выполнятся условие, все эл-ты массива суммируются

- ans+=a[i];
- out.print(ans);
- out.close();

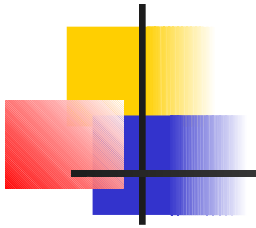
}
}

Задача E. Максимальный поток



- Имя входного файла: *input.txt*
- Имя выходного файла: *output.txt*
- Ограничение но времени: *1 second*
- Ограничение по памяти: *64 megabytes*

Лесник Иван очень любит деревья. Однажды ему потребовалось посчитать максимальный поток в дереве. Так как он не разбирается в программировании, он попросил помочь ему.



- **Формат входного файла**

В первой строке записано единственное число n $(2 \leq n \leq 100)$ — количество вершин в дереве.

В каждой из последующих $n-1$ строк записано 3 числа a, b, d $(1 \leq a, b \leq n,\ a \neq b,\ 1 \leq d \leq 1000)$ — номера вершин, соединяемых ребром и пропускная способность этого ребра.

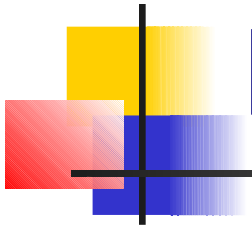
В последней строке записаны 2 числа x и y $(1 \leq x, y \leq n,\ x \neq y)$ — вершины, максимальный поток между которыми хочет посчитать лесник Иван.

- **Формат выходного файла**

Выведите одно число – максимальный поток между x и y.

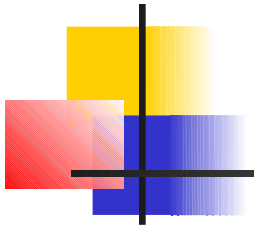
- **Пример**

input.txt	output.txt
2 1 2 100 1 2	100



Код программы

- import java.io.File;
- import java.io.IOException;
- import java.io.PrintWriter;
- import java.util.*;



- public class TaskE{

- public static void main(String[] args)throws IOException{

Scanner in = new Scanner(new File("input.txt")); *//считываем из файла*

PrintWriter out =newPrintWriter(newFile("output.txt")); *//записываем в файл*

- n = in.nextInt();

//считывает введенное число типа int n-количество вершин в дереве

- g = new int[n][n]; //двухмерный массив g

- used = new boolean[n];

//создаем массив нужного размера

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

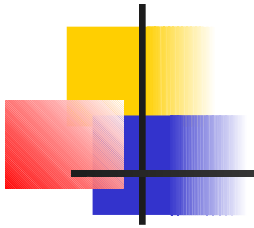
Arrays.fill(g[i], -1);

//заполняет массив g одинаковыми значениями znach = -1

ans = Integer.MAX_VALUE;

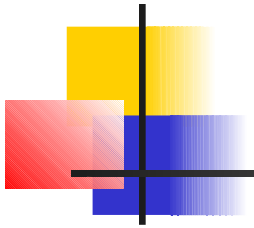
//Самая большая величина типа int

- System.out.println(ans);



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

```
            int x = in.nextInt()-1;
            int y = in.nextInt()-1;
            int w = in.nextInt();

            g[x][y] = w;
            g[y][x] = w;
        }
```

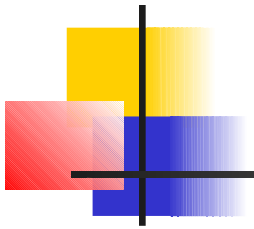


- int from = in.nextInt()-1;
  int to = in.nextInt()-1;

  dfs(from,to,ans);

  out.print(ans);
  out.close();

}



```
static int n,ans;
        static int[][] g;
        static boolean[] used;

        static void dfs(int x, int y, int max){
                used[x]=true;
                if(x==y) ans = max;

                for(int i=0; i<n; i++)
                        if(!used[i] && g[x][i]>-1)
                                dfs(i,y,Math.min(max,
   g[x][i]));
            }
}
```