Федеральное государственное автономное образовательное учреждение
высшего образования
Национальный исследовательский университет
«Высшая школа экономики»
Московский институт электроники и математики им. А.Н. Тихонова

Методические рекомендации по решению конкурсных заданий практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» в номинации «ИТ-класс» по направлению

«Программирование»

Москва, НИУ ВШЭ
2023 г.

Материалы практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» (далее – Конкурс) предназначены для оценки уровня практической подготовки участников Конкурса.

Задания практического этапа Конкурса разработаны преподавателями образовательных организаций высшего образования, участвующих в проекте «ИТ-класс в московской школе».

Индивидуальный вариант участника формируется автоматически во время проведения практического этапа Конкурса предпрофессиональных умений из базы конкурсных заданий.

Индивидуальный вариант участника включает 13 заданий.

Обобщённый план конкурсных материалов для проведения практического этапа Конкурса

№ задания	Уровень сложности	Темы элективных курсов	Контролируемые требования к проверяемым знаниям и умениям	Балл
1	Базовый	Оценка сложности алгоритмов на примере алгоритмов сортировки	Умение оценивать сложность алгоритмов. Оценка алгоритмов по памяти. Оценка работы времени алгоритма	3
2	Базовый	Элементарные структуры данных	Умение работать с элементарными структурами данных: использование списка	3
3	Базовый	Работа со строками, файлами, графикой	Умение работать с элементарными структурами данных	3
4	Базовый	Деревья поиска	Уметь корректно использовать термины, применять навыки работы с	3

			деревьями поиска	
5	Базовый	Системы контроля версий Совместная работа над проектом	Умение применять системы контроля версий. Ветки в Git. Совместная работа. Знание терминологии	3
6	Базовый	Работа со строками, файлами, графикой	Умение применять операции со строками	3
7	Повышенный	Оценка сложности алгоритмов на примере алгоритмов сортировки Работа со строками, файлами и графикой	Анализ алгоритмов с ветвлениями и циклами. Рекурсивный перебор	6
8	Повышенный	Оценка сложности алгоритмов на примере алгоритмов сортировки	Умение оценивать сложность алгоритмов. Умение применять алгоритмы сортировки	6
9	Повышенный	Элементарные структуры данных	Умение работать с элементарными структурами данных: стек, очереди, деки,	6

			использование списка	
10	Повышенный	Работа со строками, файлами и графикой	Умение решать задачи на рекурсивный перебор	6
11	Повышенный	Работа со строками, файлами и графикой	Умение применять навыки работы с графикой	6
12	Повышенный	Работа со строками, файлами и графикой	Умение решать задачи на рекурсивный перебор	6
13	Повышенный	Решение олимпиадных задач	Решение задач на графы	6

Демонстрационный вариант конкурсных заданий практического этапа Конкурса

1. Как обычно оценивают сложность алгоритма, реализованного на языке программирования? Выберите верные ответы.

1) По времени исполнения

2) По используемой памяти

3) По количеству условных конструкций

4) По количеству циклов

Теоретическая вставка: Алгоритмы характеризуются тем, что на вход могут поступать различные входные данные, которые в свою очередь преобразуются в выходные данные. Процесс работы алгоритма на входных данных имеет сложностные характеристики, например, объем используемой памяти, количество шагов алгоритма и т.д. Далее предлагается рассматривать максимальное значение по времени, то есть сложность в худшем варианте.

Например, сложностью алгоритма поиска в упорядоченном массиве из n элементов называется максимальное число сравнений элемента с элементами массива до получения ответа.

Рекомендация: ознакомиться с алгоритмами сортировки и изучить их сложность.

Решение: сложность алгоритма может оцениваться по времени исполнения и используемой памяти.

2. На алгоритмическом языке был написан алгоритм, представленный ниже:

```
алг таблица(рез цел T)
нач
цел I, цел таб Mas[1:10]
T:=0
нц для I от 1 до 10
    Mas[I]:=I+mod(I,3)
```

```
кц
нц для I от 1 до 10
        если int(Mas[I]/4)=mod(Mas[I],3)
                то T:=T+mod(I,3)
        все
кц
вывод T
кон
```

Выберите, чему равно T после третьей итерации цикла.

1) 1
2) 2
3) 34
4) 3

Решение: Сперва стоит сформировать массив Mas, который будет выглядеть следующим образом Mas = [2, 4, 3, 5, 7, 6, 8, 10, 9, 11], затем вычислить значения, которые последовательно принимает T по шагам: 0, ,2, 2, 2, 4, 4, 5, 5, 5, 6.
Легко увидеть, что на третьем шаге T принимает значение 2.

3. На алгоритмическом языке написан алгоритм, представленный ниже. Вычислите значение S после первого прохода внешнего цикла, а затем – значение S после последнего прохода внешнего цикла. Выберите, на сколько результат S после последнего прохода внешнего цикла больше, чем после первого.

```
нц для i от 1 до 2
        S:=1
        нц для j от 2 до 3
          S:=S+i+j
        кц
          S:=S+1
```

кц

1) 0
2) 1
3) 3
4) 2

Решение: поэтапно вычисляя значения S, становится очевидно, что ответ равен двум, так как после первой итерации S = 1+ (1+2) + (1+3) + 1 = 9, S = 1+ (2+2) + (2+3) + 1 = 11, после чего 11 – 9 = 2.

4. …-дерево – сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

1) АВЛ
2) CPU
3) MOD

Теоретическая вставка:

- Дерево — это структура, в которой у каждого узла может быть 0 или более подузлов. Деревья состоят из набора вершин и ориентированных ребер между ними.
- Родитель вершины f – это вершина, из которой есть ребро в f
- Дети вершины f – вершины, в которые из f есть ребро
- Лист – вершина, не имеющая потомков.
- Поддерево – вершина дерева вместе с потомками этой вершины.
- Степень вершины – количество детей.
- Глубина вершины – это расстояние от корня до вершины.
- Высота дерева – максимальная из глубин вершин дерева.
- Бинарное (двоичное) дерево – это иерархическая структура данных, в которой каждый узел имеет не более двух потомков (детей).

- АВЛ-дерево – сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

5. Выберите все корректные утверждения про системы контроля версий:

1) DVCS является полной копией CVCS
2) Фаворит среди CVCS – Python
3) В репозиториях фиксируется время изменения
4) Первыми появились CVCS

Теоретическая вставка: Системы контроля версий бывают централизованными и распределёнными.

CVCS - это более старый вид контроля версий, при котором реализуется единое хранилище версий. Разработчик вносит изменения в локальную версию, затем эти изменения реализуются в центральном репозитории. Репозиторий виден всем разработчикам, у которых есть право доступа, обмен кодом осуществляется только через центральный репозиторий.

Примеры: SVN, Perforce.

DVCS - разработчик работает с копией репозитория. Отсутствует главный репозиторий.

Примеры: git, Mercurial.

6. Перед вами представлен фрагмент кода, который был реализован на алгоритмическом языке:

```
m := Длина(s)
k = 3
s1 := Извлечь(s, k)
нц для i от 13 до m – 1
c := Извлечь(s, i)
```

s1 := Склеить(s1, c)

кц

На ввод была подана строка АФМОШЫЕКСЕНЬОРОЗНОА. Какой набор символов будет находиться в s1 после выполнения программы?

Ответ: МОРОЗНО.

Решение: m = 19, сперва извлекается строка s1, значение которой присваивается «М», после чего в цикле извлекаются посимвольно «О», «Р», «О», «З», «Н», «О» и «приклеиваются» к строке s1, поэтому после первой итерации в s1 строка «МО», после второй итерации – «МОР», и тд. После завершения программы в s1 будет строка «МОРОЗНО».

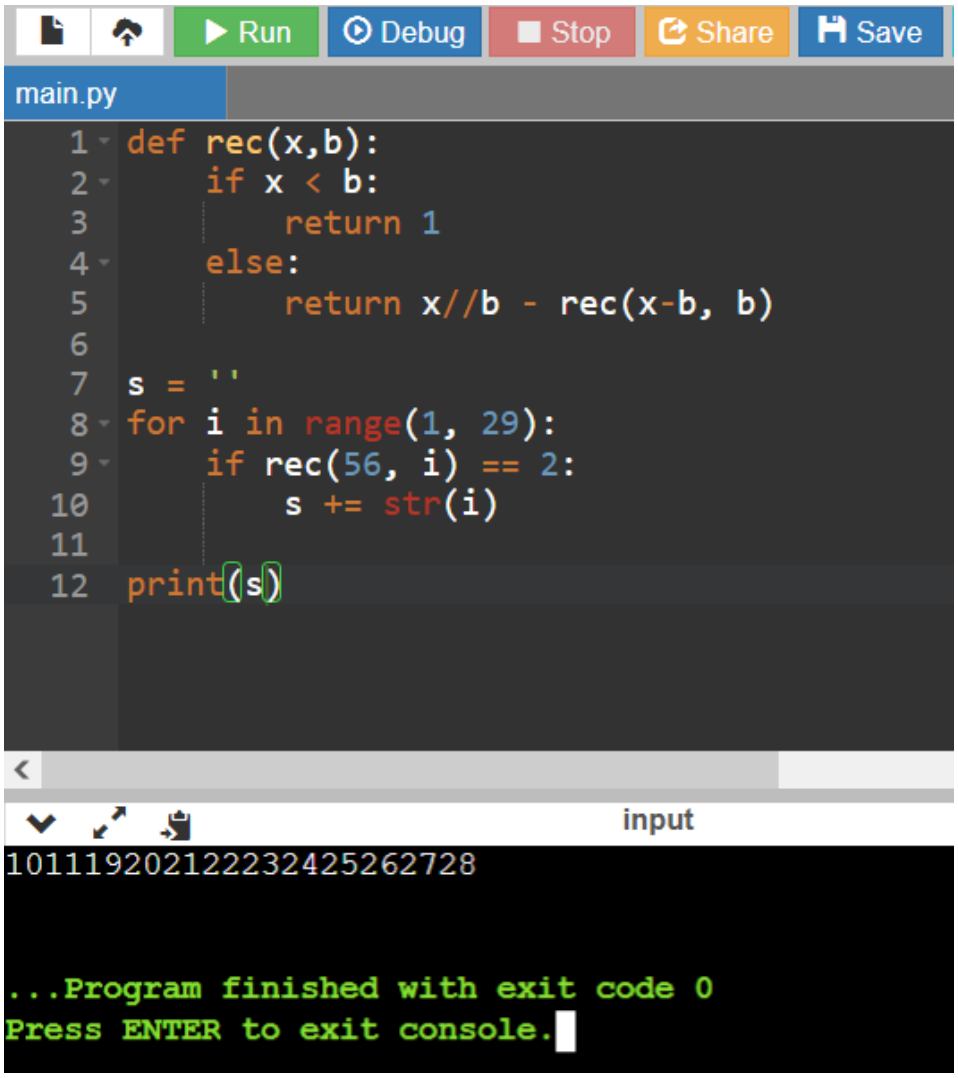
7. Перед вами представлен код рекурсивной функции. Определите, чему равно значение b, если x будет равен +56, а число, выведенное на экран, равно 2. Если есть несколько таких чисел, то запишите их слитно в порядке возрастания.

C++	**Python**

#include <iostream>
using namespace std;
int rec(int x, int b) {
  if (x < b)
  return 1;
  else
  return (x/b - rec(x-b, b));
}
int main()
{
  int x = 56;
  int b;
  cout << rec(56,b) << endl;
}
	def rec(x,b):
	  if x < b:
	  return 1
	  else:
	  return x//b - rec(x-b, b)
	
	print(rec(56,b))

Ответ: 101119202122232425262728.

Решение:



На скриншоте предложено решение задачи. Обратите внимание, что верхняя граница перебора 29, чтобы последним значением, которое примет итератор было 28, далее проверка не имеет смысла.

Программный код:

```
def rec(x,b):
    if x < b:
        return 1
    else:
```

```
        return x//b - rec(x-b, b)

s = ''
for i in range(1, 29):
    if rec(56, i) == 2:
        s += str(i)

print(s)
```

8. Зависимость времени выполнения алгоритма от количества операций для каждого из алгоритмов X и Y записана выражениями: X(n) = n ^2 + 12*n - 10, Y(n) = n ^2 + 11*n + 10, где n – это количество операций. Чему должно быть равно n, чтобы время выполнения алгоритмов стало одинаково? Ответ введите в виде числа.

Ответ: 20.

Решение: Необходимо решить квадратное уравнение n ^2 + 12*n - 10 = n ^2 + 11*n + 10. В ответ следует записать положительное значение корня. Для данного уравнения ответ равен 20.

9. Как будет выглядеть стек s после запуска фрагмента кода?

C++	**Python**

stack<int> s;
 	
s.push(1);
s.push(2);
s.pop();
s.push(2);
s.push(8);
s.pop();
	s = []
	
	def push(s, item):
	  s.append(item)
	
	def pop(s):
	  return s.pop()

s.pop();	push(s, 1) push(s, 2) pop(s) push(s, 2) push(s, 8) pop(s) pop(s)

В ответ запишите слитно числа, содержащиеся в стеке сверху вниз, НЕ используя разделителей (пробелов, запятых и т.д.).

Ответ: 1.

Решение: Стек – это тип данных, который организован по принципу LIFO, то есть последним пришел – первым ушел. В связи с этим, использованы функции добавления и удаления элементов, то есть в стек сначала добавляют 1, затем 2, затем удаляют 2, после чего добавляют 8, затем удаляют 8, затем удаляют 2, после чего остается 1.

Если бы код выглядел таким образом:

```
def pop(s):
  return s.pop()

push(s, 1)
push(s, 2)
pop(s)
push(s, 2)
push(s, 8)
```

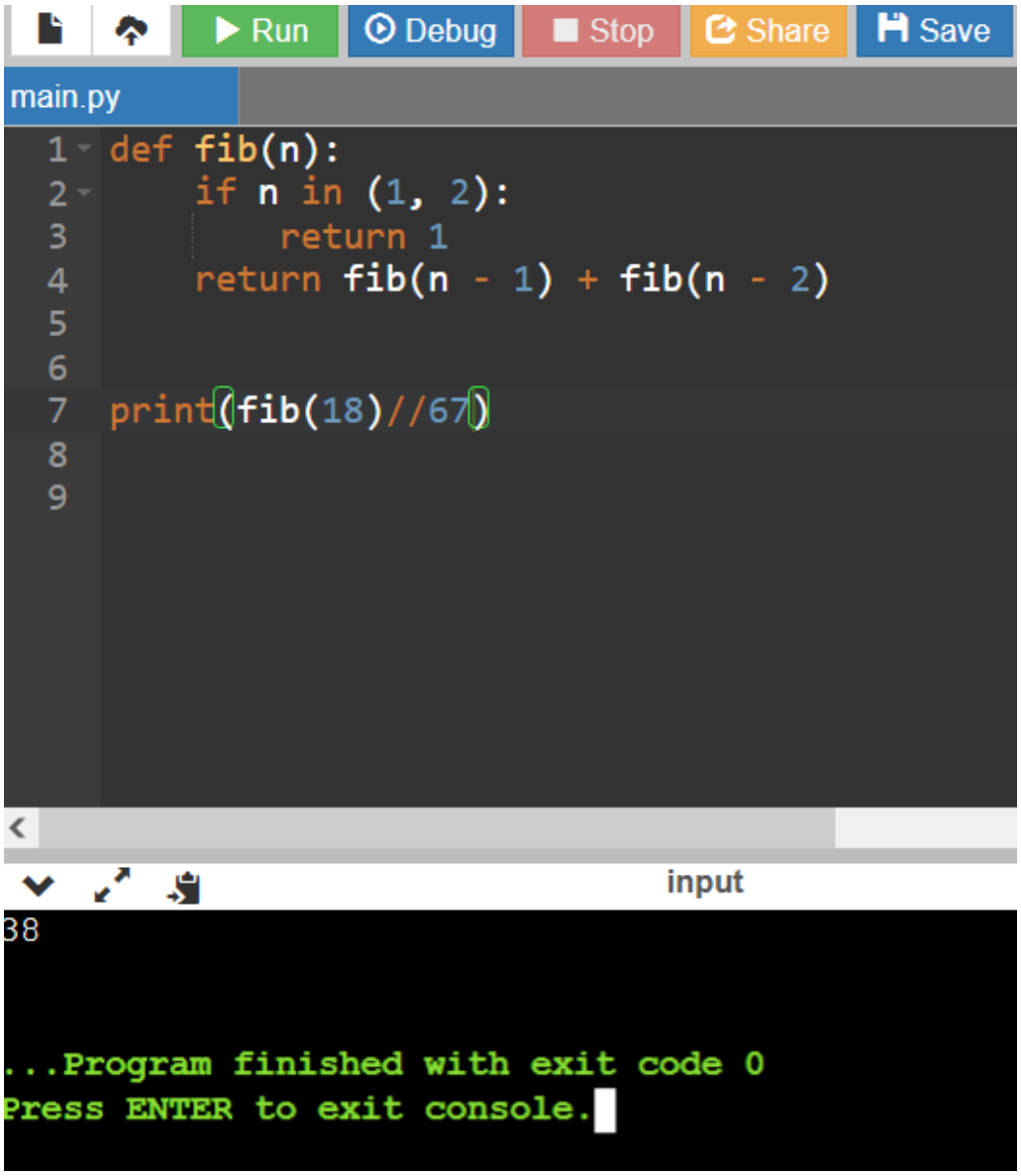
pop(s)
Тогда ответ был бы 21, так как 2 записывается в стек позже 1, то есть располагается выше (по принципу LIFO).

Если бы код выглядел следующим образом:

push(s, 1)
push(s, 2)
push(s, 2)
push(s, 8)
Тогда ответ был бы 8221, так как по принципу LIFO.

10. Найдите число Фибоначчи (f1 = 1, f2 = 1, fn = f_{n-1} + f_{n-2}), которое стоит на 18-м месте. В ответ запишите целую часть числа, которое будет получено при делении найденного числа на 67.
Ответ: 38.

Решение: Решение задачи представлено в виде программного кода, как наиболее быстрый подход к решению. Рекомендуется использовать рекурсию при реализации функции по вычислению числа Фибоначчи по его номеру. На вывод сразу подается целая часть числа при делении на 67.



Программный код:

```
def fib(n):
    if n in (1, 2):
        return 1
    return fib(n - 1) + fib(n - 2)

print(fib(18)//67)
```

11. Какой информационный объем в Мбайтах содержит файл, если было отсканировано цветное изображение размером 3x12 дюймов, разрешающая

способность сканера – 600x600 dpi, а глубина цвета – 8 бит. В ответ запишите целую часть, полученного результата.

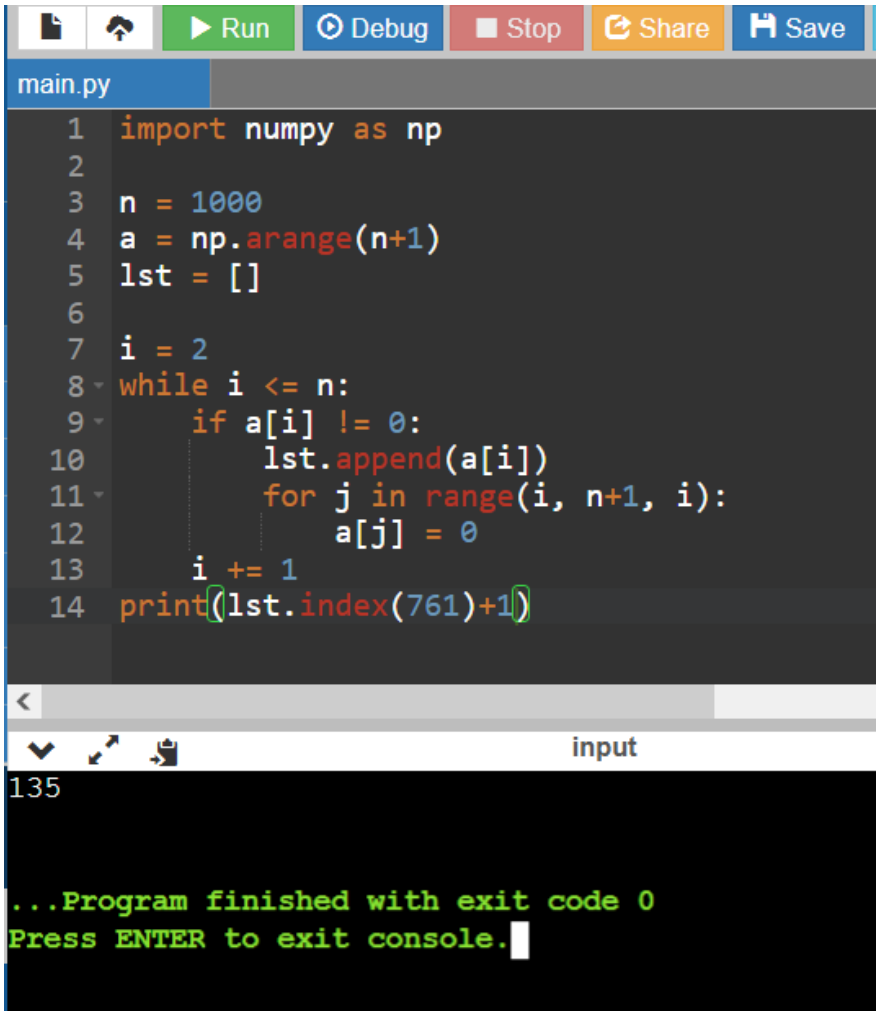
Ответ: 12.

Решение: Для расчета информационного объема цветного изображения необходимо произвести вычисления по формуле V = K*I, где V – информационный объем изображения, K – количество пикселей в изображении, I – глубина цвета.

V = 600*600*3*12*8 = 103 680 000 бит = 12 Мбайт

12. Проанализируйте список всех простых чисел в интервале от 1 до 1000. В ответ запишите число, которое соответствует индексу числа 761. Учитывайте, что индексация чисел в списке идёт с единицы.

Ответ: 135.



Программный код:

```
import numpy as np

n = 1000
a = np.arange(n+1)
lst = []

i = 2
while i <= n:
    if a[i] != 0:
        lst.append(a[i])
        for j in range(i, n+1, i):
            a[j] = 0
    i += 1
print(lst.index(761)+1)
```

С помощью алгоритма «Решето Эратосфена» формируем список из простых чисел среди чисел от 1 до 1000, после чего ищем индекс числа 761, но так как в языке программирования Python нумерация элементов идет с нуля, необходимо прибавить 1.

13. Между городами 1, 2, 3, 4, 5 существуют дороги с ухабами, в таблице указано количество ям, в которые попадает машина при выборе дороги. Если есть прочерк, то дорога разрушена полностью (проезда нет). Водителю необходимо добраться из города 2 в город 5. Какое максимальное количество ям может встретиться по дороге, посетив минимальное количество городов во время пути?

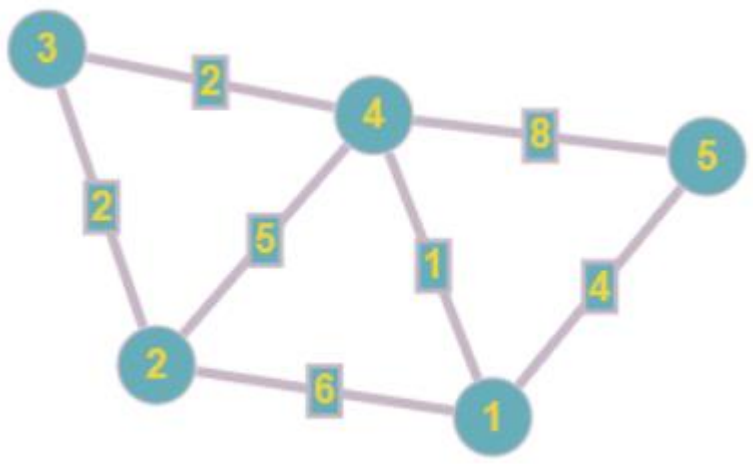
	1	2	3	4	5
1	-	6	-	1	4
2	6	-	2	5	-
3	-	2	-	2	-

4	1	5	2	-	8
5	4	-	-	8	-

1) 10
2) 9
3) 7
4) **13**

Решение:

Сперва следует построить граф



Так как стоит задача посетить минимальное количество городов, то возникает только два варианта 2-4-5 и 2-1-5, тогда2-4-5 в приоритете, так как в сумме будет 13 ям, что больше 10.