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*
*2022 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 умениям | Балл |
|---|---|---|---|---|
| 1 | Базовый | Алгоритмы и структуры данных | Умение оценивать сложность алгоритмов | 3 |
| 2 | Повышенный | Алгоритмы и структуры данных | Умение работать со строками, файлами, графикой, функциями, математическими операциями, операторами ветвления. | 6 |
| 3 | Повышенный | Алгоритмы и структуры данных | Умение оценивать сложность алгоритмов. Умение применять алгоритмы сортировки | 6 |
| 4 | Базовый | Практика программирования | Умение работать с элементарными структурами данных | 3 |
| 5 | Базовый | Практика программирования | Умение работать с элементарными структурами данных | 3 |
| 6 | Повышенный | Алгоритмы и структуры данных/Практика программирования | Умение работать со строками, файлами, графикой, списками, функциями. / Умение работать с алгоритмами поиска, деревьями поиска, хешированием, целочисленными алгоритмами. | 6 |
| 7 | Базовый | Основы проектирования программного обеспечения/Практика программирования | Жизненный цикл программного обеспечения. Качество программного обеспечения. Анализ требований к | 3 |

| | | | | |
|---|---|---|---|---|
| | | | программному обеспечению. Документирование программного обеспечения. /<br><br>Умение работать со строками, файлами, графикой. Умение осуществлять проверку и отладку программного кода. | |
| 8 | Базовый | Практика программирования | Умение применять в совместной работе над проектом системы контроля версий | 3 |
| 9 | Повышенный | Алгоритмы и структуры данных | Умение работать с алгоритмами поиска, деревьями поиска, хешированием, целочисленными алгоритмами. | 6 |
| 10 | Базовый | Алгоритмы и структуры данных | Умение работать с элементарными структурами данных. | 3 |
| 11 | Повышенный | Разработка приложений, интегрированных в ИТ-инфраструктуру | Умение применять: особенности работы приложений под управлением различных операционных систем. Защищенное хранение данных в файлах с ограниченным доступом в различных операционных системах. | 6 |
| 12 | Повышенный | Практика программирования | Умение работать со строками, файлами, графикой, функциями, математическими операциями, списками | 6 |

| | | | | |
|---|---|---|---|---|
| | | | операторами ветвления.<br>. | |
| 13 | Повышенный | Практика программирования/Основы проектирования программного обеспечения | Умение применять: Целочисленные алгоритмы. Использование связанных структур. Графы. «Жадные» алгоритмы. Алгоритм Дейкстры. Динамическое программирование. | 6 |

## Разбор демонстрационного варианта конкурсных заданий практического этапа Конкурса

**Демонстрационный вариант конкурсных заданий практического этапа Конкурса**

1. Как обычно оценивают сложность алгоритма, который реализован на каком-либо языке программирования? Может быть несколько верных пунктов.

1) По объему файла с кодом

**2) По времени исполнения**

**3) По используемой памяти**

4) По количеству строк кода

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

Рекомендация: повторить сложности алгоритмов сортировки.

2. Перед вами представлен код рекурсивной функции. Определите, сколько раз будет вызвана рекурсивная функция, если x будет равен +56, а b будет равен +6.

| **C++** | **Python** |
|---|---|
| #include <iostream><br>using namespace std;<br>int rec(int x, int b) {<br>  if (x < b)<br>  return 1;<br>  else<br>  return (x/b - rec(x-b, b));<br>}<br>int main()<br>{<br>  int i = 56;<br>  int j = 6;<br>  cout << rec(56,6) << endl; // 108} | def rec(x,b):<br>  if x < b:<br>  return 1<br>  else:<br>  return x//b - rec(x-b, b)<br><br>**print(rec(56,6))** |

Ответ: 10

Решение: В задаче представлен рекурсивный алгоритм, если пошагово рассмотреть изменения значений x и b, которые передаются в функцию, то

| x | b |
|---|---|
| 56 | 6 |

| | |
|---|---|
| 50 | 6 |
| 44 | 6 |
| 38 | 6 |
| 32 | 6 |
| 26 | 6 |
| 20 | 6 |
| 14 | 6 |
| 8 | 6 |
| 2 | 6 |

И последний 10й раз рекурсивная функция вызывается, когда x меньше b, после чего программа завершается.

Ответ:10

3. Известно, что зависимость времени выполнения алгоритма от количества операций для каждого из алгоритмов X и Y записаны выражениями: X(n) = n*n + 10, Y(n) = 12*n – 10, где n – это количество операций. Какое количество операций должен ввести Иван, чтобы время выполнения алгоритмов стало одинаково? Ответ введите в виде натурального числа.

Ответ: 10.

Решение: Необходимо решить простое уравнение:

n*n + 10 = 12*n – 10, из которого видно, что подходящим корнем будет 10, так как количество операций должно выражаться положительным числом.

Ответ: 10.

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

```
алг
нач
  целтаб Mas[1:4]
  цел k, m
  Mas[1] := 1
  Mas[2] := 27
  Mas[3] := 18
  Mas[4] := 13
  m := 0
  нц для k от 1 до 4
        если Mas[k] < 24 то
```

```
            m := m + 1
        все
  кц
  вывод m
кон
```

Выберите, чему равно m после третьей итерации цикла.

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

Решение: Таблица состоит из четырех элементов, каждый элемент в цикле проверяется, что он строго меньше 24, тогда значение переменной увеличивается на 1. После первой итерации переменная возрастает на 1, после второй не меняется, после третьей итерации m увеличивается еще на 1, поэтому после третьей итерации переменная равна 2.

Ответ: 2.

5. На алгоритмическом языке написан алгоритм, он представлен ниже. Выберите, какое значение необходимо присвоить S на старте программы, чтобы после шестой итерации цикла S было равно 71.

```
алг сумма
   вещ a,S
нач
   S:= ____;
   a:=1;
   нц
      S:=S+3*a;
      a:=a+1;
      пока a<=10
   кц
   вывод S
кон
```

**1) 8**
2) 7
3) 30

4) 15

Решение: В цикле прибавляется утроенное a, причем с каждым шагом a возрастает на единицу, то есть при первой итерации к S прибавляется 3, затем 6, затем 9, затем 12, затем 15, затем 18, то есть к стартовому значению S прибавили 63, чтобы получилось 71, S должно было быть равно 8.
Ответ: 8.

6. Как будет выглядеть стек s после запуска фрагмента кода:

| **C++** | **Python** |
|---|---|
| stack<int> s;<br><br>s.push(0);<br>s.push(2);<br>s.pop();<br>s.push(4);<br>s.push(8);<br>s.pop(); | s = []<br><br>def push(s, item):<br>  s.append(item)<br><br>def pop(s):<br>  return s.pop()<br><br>push(s, 0)<br>push(s, 2)<br>pop(s)<br>push(s, 4)<br>push(s, 8)<br>pop(s) |

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

*Ответ*: 40

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 0, затем 2, затем удаляют 2, после чего добавляют 4, 8, затем удаляют 8, поэтому сверху вниз в стеке идут 4, затем 0.
Ответ: 40.

7. … - это неконтролируемое сокращение оперативной/виртуальной памяти компьютера, которую можно устранить, исправив ошибки в программном коде.
Выберите пропущенное словосочетание, которое необходимо вставить в фразу.

**1) Утечка памяти**

2) Загруженность CPU

3) Освобождение памяти

Решение: в задании представлено определение утечки памяти.

8. Выберите два корректных утверждения про системы контроля версий:

**1) DVCS поддерживает те же опции, что и CVCS**

2) Фаворит среди CVCS – C++

**3) В репозиториях фиксируется время изменения**

4) Первыми появились DVCS

Решение: системы контроля версий обладают некоторыми свойствами, которые указаны в пункте 1 и 3.
Рекомендация: ознакомиться с DVCS, 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.

9. Напишите программный код, который позволит вам вычислить число Фибоначчи (f1 = 1, f2 = 1, fn = $f_{n-1}$ + $f_{n-2}$ ), которое стоит на 20 месте. После того как вам станет известно число, целочисленно разделите его на 33. В ответ запишите, какое число будет получено при целочисленном делении.

Ответ: 205.

Решение:

Одно из возможных решений представлено на скриншоте:

```
f1 = 1
f2 = 1

n = 20

i = 0
while i < n - 2:
    fsum = f1 + f2
    f1 = f2
    f2 = fsum
    i = i + 1

print("Значение элемента, целочисленно деленного на 33:", f2//33)
```

input

```
Значение элемента, целочисленно деленного на 33: 205


...Program finished with exit code 0
Press ENTER to exit console.
```

Ответ: 205.

10. 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 – 2
   c := Извлечь(s, i)
   s1 := Склеить(s1, c)
кц
```

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

Ответ: ВЕСНА

Решение: m = 18, сперва извлекается строка В, после чего в цикле извлекаются посимвольно Е, С, Н, А и «приклеиваются» к строке s1, поэтому после первой итерации в s1 строка ВЕ, после второй итерации – ВЕС, и тд. После завершения программы в s1 будет строка ВЕСНА.

Ответ: ВЕСНА.

11. Запишите в ответ численное значение прав доступа в операционной системе Unix, при котором все пользователи могут читать и редактировать: (-rw-rw-rw-)

Ответ: 666

Решение: Численное значение прав доступа в операционной системе Unix, при котором все пользователи могут читать и редактировать: (-rw-rw-rw-) - это 666.

12. Напишите алгоритм, который позволяет получить список всех простых чисел в интервале от 1 до 1000. В ответ запишите число, которое получается при умножении 135-го простого числа на 5.

Ответ: 3805

**Решение:**

```
1  import numpy as np
2
3  #Решето Эратосфена
4  n = 1000
5  a = np.arange(n+1)
6  lst = []
7
8  i = 2
9  while i <= n:
10     if a[i] != 0:
11         lst.append(a[i])
12         for j in range(i, n+1, i):
13             a[j] = 0
14     i += 1
15 print(lst[134]*5) #135й элемент умножается на 5
```

```
3805


...Program finished with exit code 0
Press ENTER to exit console.
```

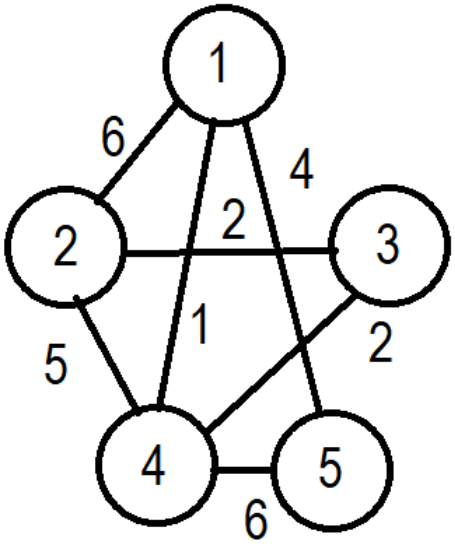
**Ответ: 3805.**

13. Между королевствами 1, 2, 3, 4, 5 существуют пути с препятствиями, в таблице указано количество подвигов, которое требуется совершить для преодоления пути между королевствами, если стоит прочерк, то путь разрушен. Юный герой решает добраться из королевства 2 в королевство 5. Какое максимальное количество подвигов он сможет совершить, пройдя самый короткий путь?

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

1) 10
2) 9
3) 7
4) **11**

Решение: Проще всего решить задачу, нарисовав граф, соответствующий таблице:




Из 2 в 5 можно попасть несколькими способами без повторения вершин, например, 2-3-4-5, 2-1-4-5, 2-1-5, 2-4-5. Самые короткие пути, по которым может пройти герой – это 2-1-4 и 2-4-5, больше всего подвигов герой сможет совершить, если пройдет по пути 2-4-5 и совершит 5+6 = 11 подвигов.

Ответ: 11 подвигов.