Работа призера заключительного этапа

командной инженерной олимпиады школьников

Олимпиада Национальной технологической инициативы

Профиль «ТЕХНОЛОГИИ БЕСПРОВОДНОЙ СВЯЗИ»

Ильина Елизавета Евгеньевна

Класс: 8

Город: Красноярск

Школа: МБОУ СШ №143

Регион: Красноярский край

Уникальный номер участника: 143

Команда на заключительном этапе: Огонь

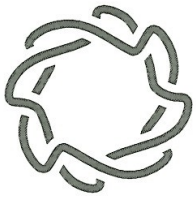
Параллель: 9-11 класс

Результаты заключительного этапа:

	Индивидуальная часть											Командная часть		Результат
	Математика				Информатика						Итого	Макс. балл	Инж. задача	Итого	
№	1	2	3	4	1.1	1.2	2.1	2.2	3.1	3.2					
143	15	0	25	0	7	0	0	-	-	-	200			3,613	65,225

Индивидуальная часть

Персональный лист участника с номером 143:



Олимпиада НТИ

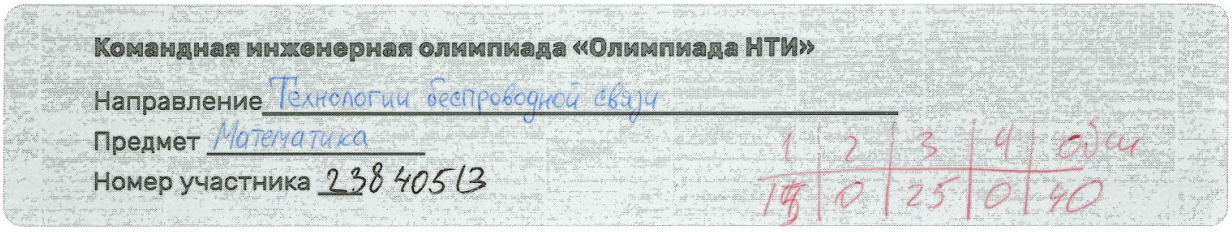
ФИО Ильина Елизавета Евгеньевна

Город Красноярск

Школа № 143

Математика

Лист 1:


Командная инженерная олимпиада «Олимпиада НТИ»
Направление Технологии беспроводной связи
Предмет Математика
Номер участника 2384053

1	2	3	4	Общ
15	0	25	0	40

Задание № 1.

1) $x=1$ и $y=0$ $1-1\cdot0+0=1$
2) $x=0$ и $y=1$ $0-0\cdot1+1=1$
3) x = любые числа (и положительные, и отрицательные), $y=1$

Например:

- $x=-1$ и $y=1$
  $-1+1+1=1$
- $100-100\cdot1+1=1$
- $25{,}5-25{,}5\cdot1+1=1$
- $-2{,}8-72{,}8\cdot1+1=1$

15 – верный ответ и решение

Оценка за задачу №1: 15

Комментарий к решению: задача решена верно.

Лист 2:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Технологии беспроводной связи

Предмет Математика

Номер участника 23840513

Задание №2.

Я считаю, что такое не могло быть. Потому что в данном роже не одновременно более интересные задачи и более высокая зарплата. И следовательно, только решать интересные задачи или только получать высокую зарплату ~~ет~~ кандидат не сможет.

0 – неправильный ответ, неверный ход рассужд.

Оценка за задачу №2: 0

Комментарий к решению: задача решена неверно.

Лист 3:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Технологии беспроводной связи

Предмет Математика

Номер участника 2384053

Задание №3.

$\frac{11110}{11111} \rightarrow \frac{44441}{44445} \rightarrow \frac{33331}{33334}$

Так как:

$\frac{11110}{11111} = 0,9090900909090...$

$\frac{44441}{44445} = 0,9100012...$

$\frac{33331}{33334} = 0,9100017...$

25 – правильный ответ, неполное решение

Оценка за задачу №3: 25

Комментарий к решению: верный ответ, не показан ход решения.

Лист 4:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Технологии беспроводной связи

Предмет Математика

Номер участника 23840513

Задание №4.

Я думаю, что 7 маршрутов могут быть в городе

0 – неправильный ответ, нет решения

Оценка за задачу №4: 0

Комментарий к решению: задача решена неверно.

Информатика

Задача 1.1:

```
lineIn=''
lineIn = input()
#lineIn = lineIn.strip()
m = lineIn.split(' ')
#print(m)
digits = []
for symbol in range (int(m[0])):
#    print(symbol)
    lineIn1 = input()
#    lineIn1 = lineIn1.strip()
    m1 = lineIn1.split(' ')
    digits.append(m1)
#    print(m1)
#print(digits)

for symbol in range (int(m[1])):
    #print(symbol)
    otstup=0
    lineIn1 = input()
    lineIn1 = int(lineIn1)
    #print('-----')
    #print(lineIn1)
    for i in range(int(m[0])):
        #print(lineIn1)
        #print(digits[i][0])
        if lineIn1 == int(digits[i][0]):
            print(otstup)
            otstup=0
        else:
            otstup=otstup+int(digits[i][1])


#    lineIn1 = lineIn1.strip()
#    m1 = lineIn1.split(' ')
#    digits.append(m1)
 #   print(m1)
```

Оценка за задачу №1.1: 7

Комментарий к решению: задача решена верно.

Задача 1.2:

```
lineIn=''
lineIn = input()
#lineIn = lineIn.strip()
m = lineIn.split(' ')
#print(m)
digits = []
for symbol in range (int(m[0])):
#    print(symbol)
    lineIn1 = input()
#    lineIn1 = lineIn1.strip()
    m1 = lineIn1.split(' ')
    digits.append(m1)
#    print(m1)
#print(digits)

for symbol in range (int(m[1])):
    #print(symbol)
    otstup=0
    lineIn1 = input()
    lineIn1 = int(lineIn1)
    #print('-----')
    #print(lineIn1)
    for i in range(int(m[0])):
        #print(lineIn1)
        #print(digits[i][0])
        if lineIn1 == int(digits[i][0]):
            print(otstup)
            otstup=0
        else:
            otstup=otstup+int(digits[i][1])


#    lineIn1 = lineIn1.strip()
#    m1 = lineIn1.split(' ')
#    digits.append(m1)
 #   print(m1)
```

Оценка за задачу №1.2: 0

Комментарий к решению: задача решена неверно.

Задача 2.1:

```
# put your python code here
lineIn=''
lineIn = input()
#lineIn = lineIn.strip()
m = lineIn.split(' ')
#print(m)
digits = []
for symbol in range (int(m[0])):
#    print(symbol)
    lineIn1 = input()
#    lineIn1 = lineIn1.strip()
    m1 = lineIn1.split(' ')
    digits.append(m1)
#    print(m1)
#print(digits)

lend = len(digits)
if len(lineIn)>0:
    for i in range (lend):
        k0=int(digits[i][0])
        k1=int(digits[i][1])
        digits[i][0] = k1
        digits[i][1] = k0
#print(digits)
digits.sort(reverse=True)
#print(digits)
sum=0
tim=0
for i in range (lend):
    tim= tim + digits[i][1]
    if tim <= int(m[1]):
        sum = sum + digits[i][0]
print(sum)
```

Оценка за задачу №2.1: 0

Комментарий к решению: задача решена неверно.

Командная часть

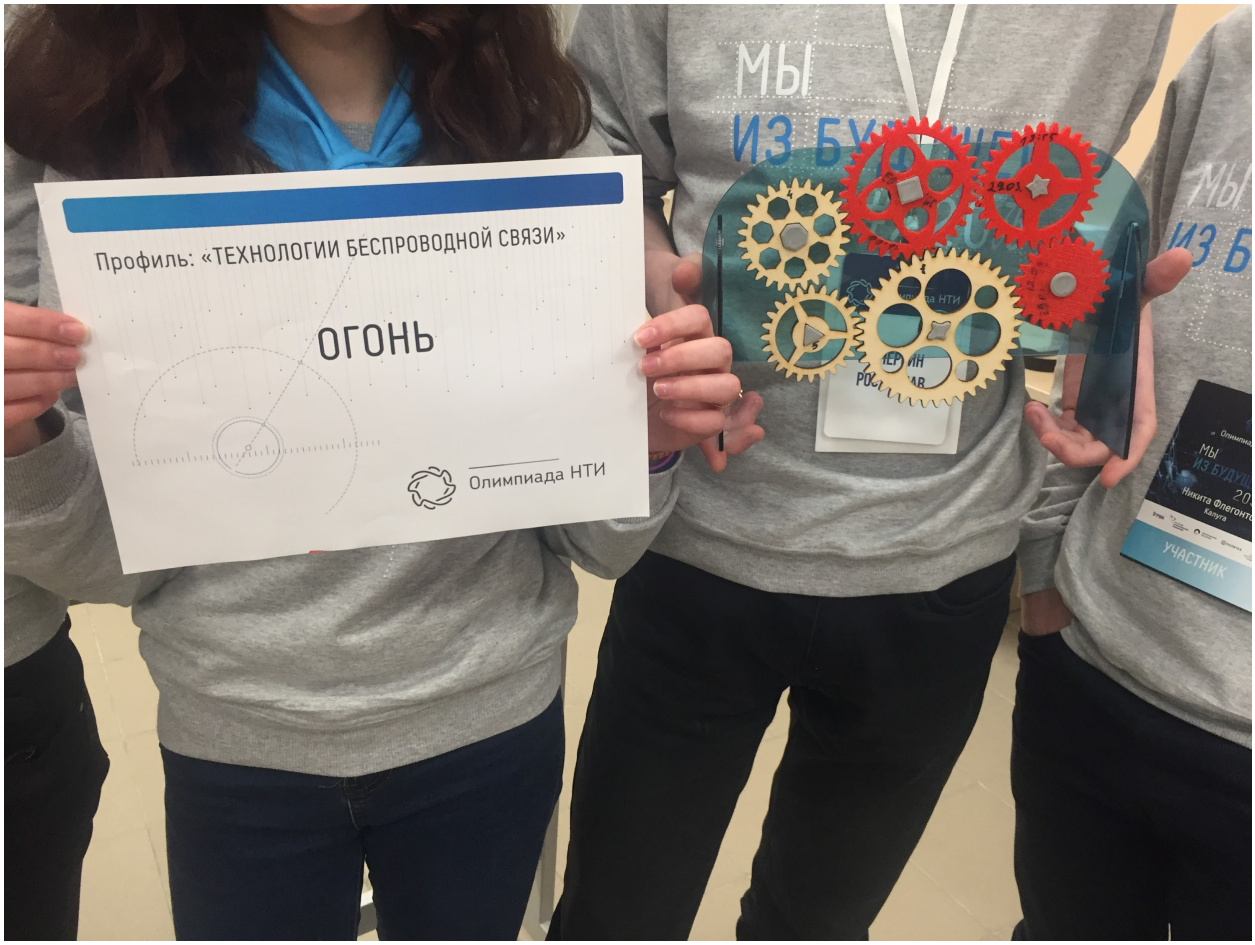
Результаты были получены в рамках выступления команды: Огонь

Личный состав команды:

- Густырь Валерия Александровна
- Епишина Анастасия Витальевна
- Ильина Елизавета Евгеньевна
- Кочергин Ростислав Алексеевич
- Флегонтов Никита
- Чемерев Илья Денисович

Описание решений командной задачи заключительного этапа:

1. Фото результата:



2. Код решения:

h_utils.py

```
import random

# длина блока кодирования
CHUNK_LENGTH = 8

# проверка длины блока
assert not CHUNK_LENGTH % 8, 'Длина блока должна быть кратна 8'

# вычисление контрольных бит
CHECK_BITS = [i for i in range(1, CHUNK_LENGTH + 1) if not i & (i - 1)]


def to_bin(chars):
    """
    Преобразование символов в бинарный формат
    """
    return ''.join([bin(byte)[2:].zfill(8) for byte in chars])


def chunk_iterator(text_bin, chunk_size=CHUNK_LENGTH):
    """
    Поблочный вывод бинарных данных
    """
    for i in range(len(text_bin)):
        if not i % chunk_size:
            yield text_bin[i:i + chunk_size]


def get_check_bits_data(value_bin):
    """
    Получение информации о контрольных битах из бинарного блока данных
    """
    check_bits_count_map = {k: 0 for k in CHECK_BITS}
    for index, value in enumerate(value_bin, 1):
        if int(value):
            bin_char_list = list(bin(index)[2:].zfill(8))
            bin_char_list.reverse()
            for degree in [2 ** int(i) for i, value in enumerate(bin_char_list) if int(value)]:
                check_bits_count_map[degree] += 1
    check_bits_value_map = {}
    for check_bit, count in check_bits_count_map.items():
        check_bits_value_map[check_bit] = 0 if not count % 2 else 1
    return check_bits_value_map
```

```
def set_empty_check_bits(value_bin):
    """
    Добавить в бинарный блок "пустые" контрольные биты
    """
    for bit in CHECK_BITS:
        value_bin = value_bin[:bit - 1] + '0' + value_bin[bit - 1:]
    return value_bin


def set_check_bits(value_bin):
    """
    Установить значения контрольных бит
    """
    value_bin = set_empty_check_bits(value_bin)
    check_bits_data = get_check_bits_data(value_bin)
    for check_bit, bit_value in check_bits_data.items():
        value_bin = '{0}{1}{2}'.format(
            value_bin[:check_bit - 1], bit_value, value_bin[check_bit:])
    return value_bin


def get_check_bits(value_bin):
    """
    Получить информацию о контрольных битах из блока бинарных данных
    """
    check_bits = {}
    for index, value in enumerate(value_bin, 1):
        if index in CHECK_BITS:
            check_bits[index] = int(value)
    return check_bits


def exclude_check_bits(value_bin):
    """
    Исключить информацию о контрольных битах из блока бинарных данных
    """
    clean_value_bin = ''
    for index, char_bin in enumerate(list(value_bin), 1):
        if index not in CHECK_BITS:
            clean_value_bin += char_bin

    return clean_value_bin


def check_and_fix_error(encoded_chunk):
```

```
    """
    Проверка и исправление ошибки в блоке бинарных данных
    """
    check_bits_encoded = get_check_bits(encoded_chunk)
    check_item = exclude_check_bits(encoded_chunk)
    check_item = set_check_bits(check_item)
    check_bits = get_check_bits(check_item)
    if check_bits_encoded != check_bits:
        invalid_bits = []
        for check_bit_encoded, value in check_bits_encoded.items():
            if check_bits[check_bit_encoded] != value:
                invalid_bits.append(check_bit_encoded)
        num_bit = sum(invalid_bits)
        encoded_chunk = '{0}{1}{2}'.format(
            encoded_chunk[:num_bit - 1],
            int(encoded_chunk[num_bit - 1]) ^ 1,
            encoded_chunk[num_bit:])
    return encoded_chunk


def get_diff_index_list(value_bin1, value_bin2):
    """
    Получить список индексов различающихся битов
    """
    diff_index_list = []
    for index, char_bin_items in enumerate(zip(list(value_bin1), list(value_bin2)), 1):
        if char_bin_items[0] != char_bin_items[1]:
            diff_index_list.append(index)
    return diff_index_list


def encode(source):
    """
    Кодирование данных
    """
    text_bin = to_bin(source)
    result = ''
    for chunk_bin in chunk_iterator(text_bin):
        chunk_bin = set_check_bits(chunk_bin)
        result += chunk_bin
    return result


def decode(encoded, fix_errors=True):
    """
```

```
    Декодирование данных
    """
    fixed_encoded_list = []
    for encoded_chunk in chunk_iterator(encoded, CHUNK_LENGTH +
len(CHECK_BITS)):
        if fix_errors:
            encoded_chunk = check_and_fix_error(encoded_chunk)
        fixed_encoded_list.append(encoded_chunk)

    clean_chunk_list = []
    for encoded_chunk in fixed_encoded_list:
        encoded_chunk = exclude_check_bits(encoded_chunk)
        clean_chunk_list.append(encoded_chunk)
    ans = bytearray(b"")
    for clean_chunk in clean_chunk_list:
        for clean_char in [clean_chunk[i:i + 8] for i in range(len(clean_chunk)) if not i %
8]:
            ans.append(int(clean_char, 2))
    return ans
```

humming_encode.py

```
import h_utils


def byte_to_char(s):
    c = chr(int(s, 2))
    return c

inp = open("gear6.7z", "br")
outp = open("encoded.txt", "bw")


cur = inp.read(2)
while cur:
    byt = bytearray(b"")
    if len(cur) == 1:
        cur = b"\x00"
    st = h_utils.encode(cur)
    byt.append(int(st[:8], 2))
    byt.append(int(st[8:16], 2))
    byt.append(int(st[16:24], 2))
    cur = inp.read(2)
    outp.write(byt)
```

```
outp.close()
inp.close()
```

humming_decode.py

```
import h_utils


inp = open("encoded_inv.txt", "rb")
outp = open("out.7z", "wb")

cur = inp.read(3)

while len(cur):
    out = h_utils.decode(h_utils.to_bin(cur))
    outp.write(out)
    cur = inp.read(3)

inp.close()
outp.close()
```

invert.py

```
inp = open("encoded1.txt", "br")
outp = open("encoded_inv.txt", "bw")

fin = inp.read(1024)

switch = 0

while fin:
    if switch == 0:
        outp.write(fin)
    else:
        out = bytearray(b"")
        for byte in fin:
            out.append(255 - int(byte))
        outp.write(out)
    fin = inp.read(1024)
    switch = (switch + 1) % 2

outp.close()
inp.close()
```

3. Протокол действий

Имя файла	Размер, KB	MD5	Имя сохр. файла	MD5 сохр. файла	Начало	Окончание	Длительность	Скорость, kbit/s
output.txt	9.9	44ebd774a49e2a7d474a6fd572d7e3d2	output.txt	09aaf02a3ca7a075315b626ecb2032f0	2017-03-26 11:38:03	2017-03-26 11:38:03	00:00:00	MAX
test.txt	0.8	2548646e95f29f68b9c2dffc19136c54	test.txt	0a2a11d278b1409b18b08674ae68ea5d	2017-03-26 12:29:57	2017-03-26 12:29:57	00:00:00	MAX
gear_6_out.txt	54.3	7d93ab89cbc53aaf9613001923767859	gear_6_out.txt	343835cb06dad2cac084b555ffd2e3a1	2017-03-26 14:11:47	2017-03-26 14:11:54	00:00:07	62.1
gear_6_oneln.txt	54.3	5a3eda3598a17d407e916e52d8412673	gear_6_oneln.txt	6419f4765ec60530e0c8594cb039ea46	2017-03-26 15:07:39	2017-03-26 15:07:46	00:00:07	62.1
gear_6_ml.txt	69.0	f5a2c395e8096852bff89fc749ef030f	gear_6_ml.txt	bde66603fbe79b219c8fa4a358a3ad9e	2017-03-26 15:07:55	2017-03-26 15:08:04	00:00:09	61.3
gear_6_oneln — 3.txt	54.3	5a3eda3598a17d407e916e52d8412673	gear_6_oneln — 3.txt	417023f771ab16b849239f106066b034	2017-03-26 15:32:43	2017-03-26 15:3	00:00:15	29.0

						2:58		
gear_6_oneln — 2.txt	54.3	5a3eda3598a17d407e916e52d8412673	gear_6_oneln — 2.txt	b0acc0289b7d4d723e862a02f27fc9d6	2017-03-26 15:33:05	2017-03-26 15:33:21	00:00:16	27.1
gear_6_oneln — 1.txt	54.3	5a3eda3598a17d407e916e52d8412673	gear_6_oneln — 1.txt	b61e508c0e8708c7b7c2aa03cef1b466	2017-03-26 15:33:36	2017-03-26 15:33:51	00:00:15	29.0
gear_6_ml — 3.txt	69.0	f5a2c395e8096852bff89fc749ef030f	gear_6_ml — 3.txt	01a9612d82c6bdb852c42142c848f713	2017-03-26 15:34:01	2017-03-26 15:34:20	00:00:19	29.1
gear_6_ml — 2.txt	69.0	f5a2c395e8096852bff89fc749ef030f	gear_6_ml — 2.txt	3b96037d8f40884ed1077ca34c99f678	2017-03-26 15:34:29	2017-03-26 15:34:47	00:00:18	30.7
gear_6_ml — 1.txt	69.0	f5a2c395e8096852bff89fc749ef030f	gear_6_ml — 1.txt	a3a462ba645763c7b0365f61d792fa9f	2017-03-26 15:34:57	2017-03-26 15:35:17	00:00:20	27.6
gear_6_hout.txt	790.3	a898741268f325ea0816ba9ff91955b9	gear_6_hout.txt	b715225ff8238b97d8c8fa61786dc3c6	2017-03-27 17:15:18	2017-03-27 17:17:01	00:01:43	61.4
gear_6_hout.bin	790.3	a898741268f325ea0816ba9ff91955b9	gear_6_hout.bin	52c4232f4d9dbf744fbd8cf39fcbdc77	2017-03-27 17:41:04	2017-03-27 17:42:46	00:01:42	62.0
gear_	127.1	92cd51620d8de	gear_	f1eb06d97ac74	2017-	201	00:	63.5

6_hout.txt		b03ec19875185dc1062	6_hout(1).txt	804d0d541d8526cd8c5	03-28 15:05:24	7-03-28 15:05:40	00:16	
gear_6_fout.txt	98.7	e3284614135a64c8cfbab92ef3a7a58d	gear_6_fout.txt	9f6e03ea7d634d7753aaa891757d2375	2017-03-28 16:41:09	2017-03-28 16:41:22	00:00:13	60.7
gear_6_fout.txt	65.9	6f23abbc78fcd4ad57d0780be559aa9d	gear_6_fout(1).txt	20ea0391758af925f1190e781060b64c	2017-03-28 17:08:55	2017-03-28 17:09:04	00:00:09	58.6
gear_6_fout.txt	65.9	afdfc4da2b34bab910b9937776997301	gear_6_fout(2).txt	4452e4726556cca8cd9b37e2152a0497	2017-03-28 17:49:26	2017-03-28 17:49:35	00:00:09	58.6
gear_6.zip	53.7	fadc88b36de8817037273fbc64729e09	gear_6.zip	9b0ac73b6d14feba5a3ae3c0a3cf42d6	2017-03-29 10:23:38	2017-03-29 10:23:53	00:00:15	28.6
encoded.txt	11.1	5b00adfcb8214c4cec49fcecafae5cf1	encoded.txt	e51711e5e0244d3b525dad6a39df7434	2017-03-29 11:26:57	2017-03-29 11:26:57	00:00:00	MAX
encoded.txt	11.1	5b00adfcb8214c4cec49fcecafae5cf1	encoded(1).txt	6e7a04aff9e1ad1d1f1db698c24bf6d4	2017-03-29 11:34:55	2017-03-29 11:34:55	00:00:00	MAX
encoded.txt	20.3	12752fb8452fb820ea707ae427ed43ef	encoded(2).txt	8bd784769dd5cc8ba5c3d8fda59c7232	2017-03-29 12:35:	2017-03-	00:00:01	162.4

					25	29 12:35:26		
encoded.txt	27.7	90025092682c2ec6a70a653c840b51dd	encoded(3).txt	1616bade058dd77f1c4ffc26fe1ff2ca	2017-03-29 13:44:29	2017-03-29 13:44:32	00:00:03	73.9
encoded.txt	27.8	011146c118ce20efd25718b35abb5fc0	encoded(4).txt	eecbc276b49335966cc84e3a8ebcc079	2017-03-29 14:04:00	2017-03-29 14:04:03	00:00:03	74.1

Общее время: 00:06:40