24 задание – 3 балла

- Внимательно читайте задание (привести пример трехзначного числа
  или только четного числа)
- Выписывайте строку с ошибкой и строку с исправлением:
  Ошибка 1: Ошибка 2:
  Исправление 1: Исправление 2:
- НЕ правильно найденная строка: - 1 балл!
- Нужно найти алгоритмические ошибки!!! (не правильно поставленная ;
  описание типов – это не ошибки!!!)
- Языки СИ++, Python – аккуратно с синтаксисом!!! (= или ==)

Требовалось написать программу, при выполнении которой с клавиатуры считывается натуральное число N, не превосходящее 10^9, и выводится сумма цифр этого числа. Программист написал программу неправильно.

Паскаль	Python	Си

var N: longint;
  sum, d: integer;
begin
  readln(N);
  sum := 1;
  while N > 0 do begin
  d := N mod 10;
  N := N div 10;
  sum := d;
  end;
  writeln(sum);
end.
	N = int(input())
	sum = 1
	while N > 0:
	  d = N % 10
	  N = N // 10
	  sum = d
	print(sum)
		#include <stdio.h>
		int main()
		{
		  int N, sum, d;
		  scanf("%d", &N);
		  sum = 1;
		  while (N > 0) {
		  d = N % 10;
		  N = N / 10;
		  sum = d;
		  }
		  printf("%d", sum);
		  return 0;
		}

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 256.
2. Приведите пример такого трёхзначного числа X, при вводе которого программа выведет правильный ответ.
3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:
   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Дано натуральное число N, не превосходящее 10^8. Необходимо найти и вывести число, которое получится при записи N справа налево и удалении всех единиц. Ведущие нули выводить не надо. Если в числе N нет цифр кроме единиц и нулей, необходимо вывести 0. Например, при вводе числа 1984 нужно вывести 489, а при вводе 2001 нужно вывести 2. Программист

Паскаль	Python	Си

var n, m: longint;
d: integer;
begin
read(n);
m := 0;
while n>=1 do begin
d := n mod 10;
if d > 1 then begin
m := 10*d + m;
end;
n:= (n - d) div 10;
end;
write(m);
end.
	n = int(input())
	m = 0;
	while n >= 1:
	d = n % 10
	if d > 1:
	m = 10*d + m;
	n = (n - d) // 10;
	print(m)
		#include <stdio.h>
		int main()
		{
		int n, sum, d;
		scanf("%d", &n);
		m = 0;
		while (n >= 1) {
		d = n % 10;
		if (d > 1)
		m = 10*d + m;
		n = (n - d) / 10;
		}
		printf("%d", m);
		return 0;
		}

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 1984.
2. Приведите пример такого трёхзначного числа N, при вводе которого программа выведет правильный ответ.
3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:
   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

	N = 1984	N = 198	N = 19
d := n mod 10; m := 10*d + m;	d=4 m=10*4+m=40	d=8 m=80+40=120	d=9 m=90+120=210
n:= (n-d) **div** 10;	n=(1984-4) div 10=1980	n=(198-8) div 10=19	n=(19-9) div 10=1

На вход программы поступают 4 неотрицательных целых числа, не превышающие 1000, среди которых могут быть одинаковые. Нужно написать программу, которая выводит количество чисел, кратных 3, и максимальное из этих чисел. Если среди входных данных нет чисел, кратных трёх, программа должна вывести слово «NO». Программист написал программу неправильно.

Паскаль	Python	Си

var i, x: integer;
var maximum, count: integer;
begin
  count := 0;
  maximum := 999;
  for i:=1 to 4 do begin
  read(x);
  if x mod 3 = 0 then begin
  count := count + 1;
  if x < maximum then
  maximum := x;
  end;
  end;
  if count > 0 then begin
  writeln(count);
  writeln(maximum);
  end
  else
  writeln('NO');
end.
	count = 0;
	maximum = 999;
	for i in range(4):
	  x = int(input());
	  if x % 3 == 0:
	  count = count + 1;
	  if x < maximum:
	  maximum = x;
	if count > 0:
	  print(count);
	  print(maximum);
	else:
	  print("NO")
		#include <stdio.h>
		int main()
		{
		  int i, x;
		  int maximum, count;
		  count = 0;
		  maximum = 999;
		  for (i=1; i<= 4; i++) {
		  scanf("%d", &x);
		  if (x % 3 == 0) {
		  count = count + 1;
		  if (x < maximum)
		  maximum = x;
		  }
		  }
		  if (count > 0) {
		  printf("%d\n", count);
		  printf("%d", maximum);
		  }
		  else
		  print("NO");
		  return 0;
		}

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе чисел 2 9 4 3.
2. Приведите пример такой последовательности, содержащей число, кратное 3, при вводе которой программа выведет правильный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:

1) выпишите строку, в которой сделана ошибка;

2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

25 задание – 2 балла

- Если значения элементов массива не изменяются, а просто печатаются нужные числа – это ошибка.

Пусть k – найденная величина

Схематический пример **верного** подхода (1):

```
for i:=1 to N do begin
        if (A[i] удовлетворяет условию) then A[i]:=k;
        writeln (A[i])
end;
```

Схематический пример **верного** подхода (2):

```
for i:=1 to N do if (A[i] удовлетворяет условию) then A[i]:=k;
for i:=1 to N do writeln (A[i]);
```

Схематический пример **неверного** подхода (3):

```
for i:=1 to N do
        if (A[i] удовлетворяет условию) then writeln (k)
        else writeln (A[i]);
```

- Массив можно выводить различными способами, главное, чтобы были выведены все элементы. Например: print (a) на Python.
- Допускаются решения, использующие функциональное программирование, например, на Pascal ABC.
- За ответ на «естественном языке» или в виде блок-схемы – оценка 0.
- Отсутствие инициализации – ошибка.
- Использование переменных, не указанных в условии задания – ошибка.

Дан массив, содержащий 2019 положительных целых чисел, не превышающих 15000. Необходимо найти количество чётных элементов массива, не кратных 3, заменить все нечётные элементы, кратные 3, на это количество и вывести изменённый массив. Например, для исходного массива из пяти элементов 20, 89, 27, 92, 48 программа должна вывести числа 20, 89, 2, 92, 48.

Напишите на одном из языков программирования программу для решения этой задачи.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из описанных.

```
const
   N=2019;
var
   a: array [1..N] of integer;
   i, m, k: integer;
begin
   for i:=1 to N do
      readln(a[i]);
   ...
end.
```

В качестве ответа Вам необходимо привести фрагмент программы, который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Free Pascal 2.6). В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии.

```
const
   N=5;
var
   a: array [1..N] of integer;
   i, m, k: integer;
begin
   for i:=1 to N do
      readln(a[i]);
   k:=0;
    for i:=1 to N do
       if (A[i] mod 2=0) and (A[i] mod 3<>0) then
              k:=k+1;
     for i:=1 to N do
        if (A[i] mod 2<>0) and (A[i] mod 3=0) then
                       a[i]:=k;
    for i:=1 to N do
      writeln (A[i]);
end.
```

Дан массив, содержащий 2019 положительных целых чисел, не превышающих 15000. Необходимо найти минимальный и максимальный **чётные** элементы (если в массиве нет чётных элементов, минимум и максимум считаются равными нулю), вычислить их среднее арифметическое, уменьшить все **нечётные** элементы, превышающие это среднее, на величину этого среднего и вывести изменённый массив.

Например, для исходного массива из пяти элементов 30, 89, 27, 90, 68 программа должна вывести числа 30, 29, 27, 90, 68 (минимум и максимум равны 30 и 90, их среднее равно 60, все нечётные элементы, превышающие 60, уменьшены на 60).

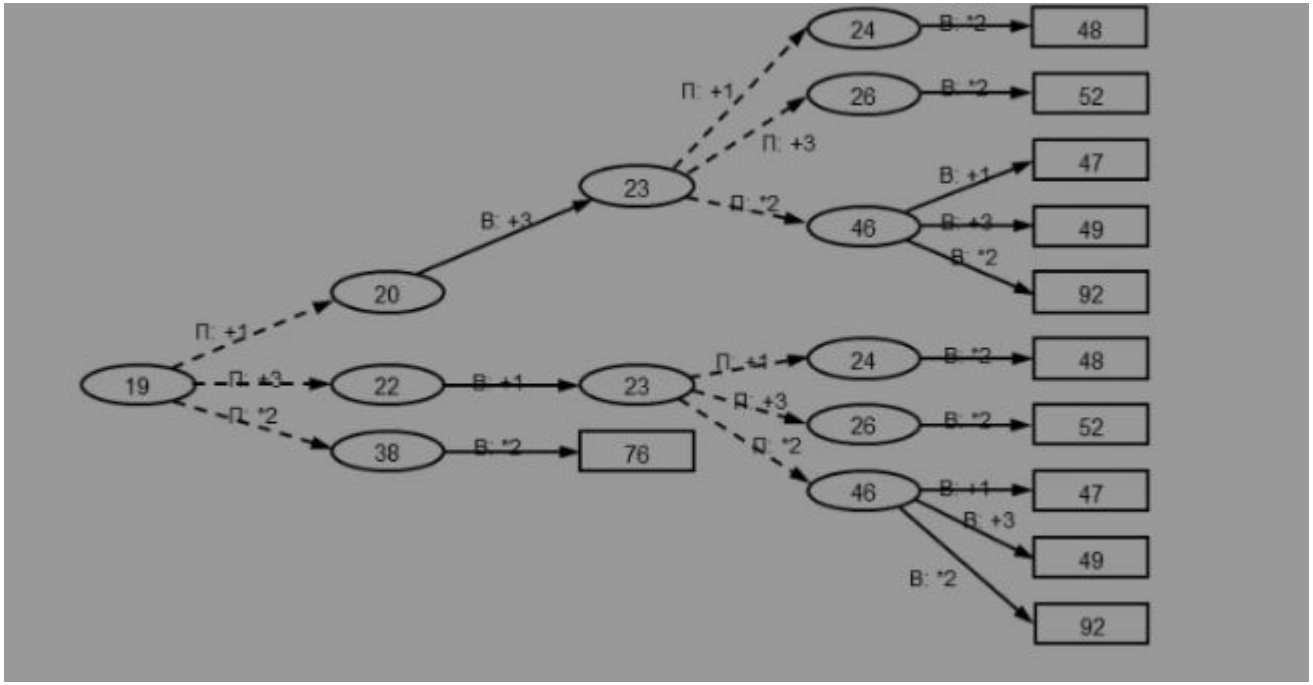
Напишите на одном из языков программирования программу для решения этой задачи.

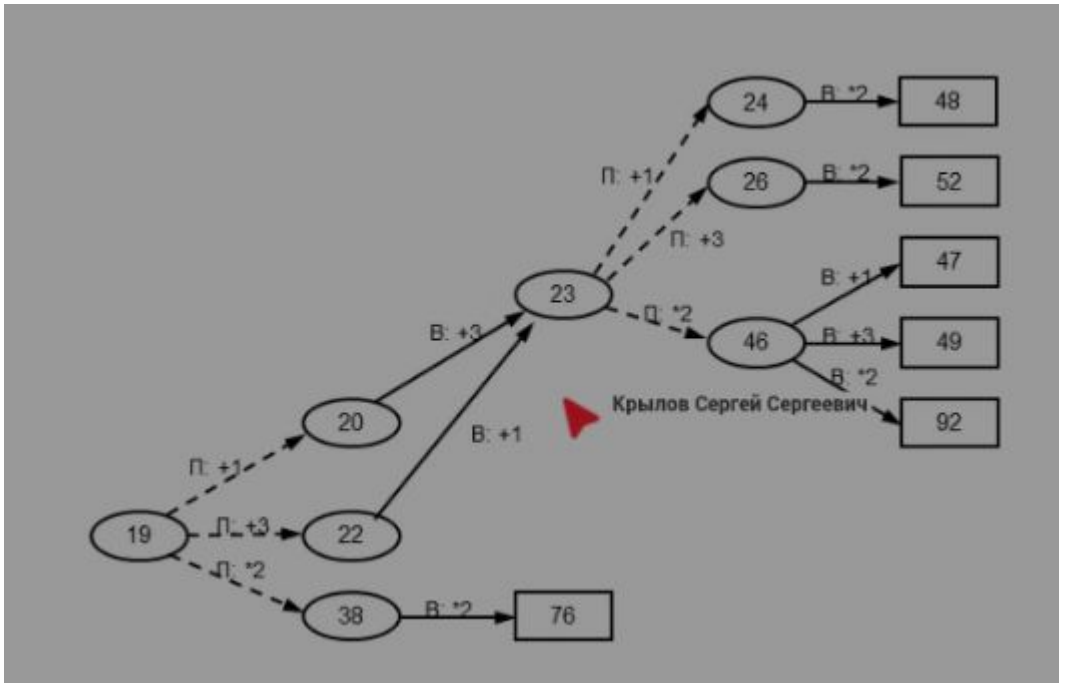
Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из описанных.

```
const
   N=2019;
var
   a: array [1..N] of integer;
   i, m, k: integer;
begin
   for i:=1 to N do
      readln(a[i]);
   ...
end.
```

В качестве ответа Вам необходимо привести фрагмент программы, который должен находиться на месте многоточия.

```
m := 15001; k := 0;
for i:=1 to N do
begin
        if (a[i] mod 2 = 0) and (a[i] < m) then m := a[i];
        if (a[i] mod 2 = 0) and (a[i] > k) then k := a[i];
end;
if m > 15000 then m := 0;
m := (m+k) div 2;
for i:=1 to N do begin
  if (a[i] mod 2 = 1) and (a[i] > m) then
    a[i] := a[i] – m;
  writeln(a[i])
end;
```


19
П: +1
П: +3
П: *2
20
22
38
В: +3
В: +1
В: *2
23
76
П: +1
П: +3
П: *2
24
26
46
В: *2
В: *2
В: +1
В: +3
В: *2
48
52
47
49
92
П: +1
П: +3
П: *2
24
26
46
В: *2
В: *2
В: +1
В: +3
В: *2
48
52
47
49
92


19
П: +1
П: +3
П: *2
20
22
38
В: +3
В: +1
В: *2
23
76
П: +1
П: +3
П: *2
24
26
46
В: *2
В: *2
В: +1
В: +3
В: *2
48
52
47
49
92
Крылов Сергей Сергеевич

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в одну из куч (по своему выбору) **один** камень или увеличить количество камней в куче в **три раза**. Например, пусть в одной куче 10 камней, а в другой 7 камней; такую позицию в игре будем обозначать (10, 7). Тогда за один ход можно получить любую из четырёх позиций:
(11, 7), (30, 7), (10, 8), (10, 21). Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней.
Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 68. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, при которой в кучах будет 68 или больше камней.
В начальный момент в первой куче было 6 камней, во второй куче – S камней; $1 \leq S \leq 61$.

Будем говорить, что игрок имеет *выигрышную стратегию*, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника. В описание выигрышной стратегии **не следует** включать ходы играющего по этой стратегии игрока, не являющиеся для него безусловно выигрышными, т.е. не являющиеся выигрышными независимо от игры противника.

Выполните следующие задания.

Задание 1

в) Укажите все такие значения числа S, при которых Петя может выиграть за один ход.

г) Известно, что Ваня выиграл своим первым ходом после неудачного первого хода Пети. Укажите минимальное значение S, когда такая ситуация возможна.

Задание 2

Укажите такое значение S, при котором у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:

- Петя не может выиграть за один ход;
- Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Задание 3

Укажите значение S, при котором одновременно выполняются два условия:

- у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
- у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Для указанного значения S опишите выигрышную стратегию Вани.

Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы).

В узлах дерева указывайте позиции, на рёбрах рекомендуется указывать ходы. Дерево не должно содержать партии, невозможные при реализации выигрывающим игроком своей выигрышной стратегии. Например, полное дерево игры не является верным ответом на это задание.

6

$1 \le S \le 61.$

+1
*3

Завершение игры
>=68

1.
а)
6+3*20=66
6+3*21=68!

S∈ [21; 61]

Б) s=20, s=7
Min = 7

Задание 1

а) Петя может выиграть при $21 \le S \le 61$.
б) $S = 7$.

6

$1 \le S \le 61.$

+1
*3

Завершение игры
>=68

Задание 2

Возможное значение S: 20. В этом случае Петя, очевидно, не может выиграть первым ходом. Однако он может получить позицию (7, 20). После хода Вани может возникнуть одна из четырёх позиций: (8, 20), (21, 20), (7, 21), (7, 60). В каждой из этих позиций Петя может выиграть одним ходом, утроив количество камней во второй куче.

Замечание для проверяющего. Ещё одно возможное значение S для этого задания – число 13. В этом случае Петя первым ходом должен утроить количество камней в меньшей куче и получить позицию (6*3, 13) = (18, 13). При такой позиции Ваня не может выиграть первым ходом, а после любого хода Вани Петя может выиграть, утроив количество камней в большей куче. Достаточно указать одно значение S и описать для него выигрышную стратегию.

S=20
(6,20)

+1 Петя 1 ход

(7,20)

(8,20) (7,21) (21,20) (7,60)

+1 Ваня 1 ход

Петя *3 и выигрывает вторым ходом

Ваня) выделены жирным шрифтом. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

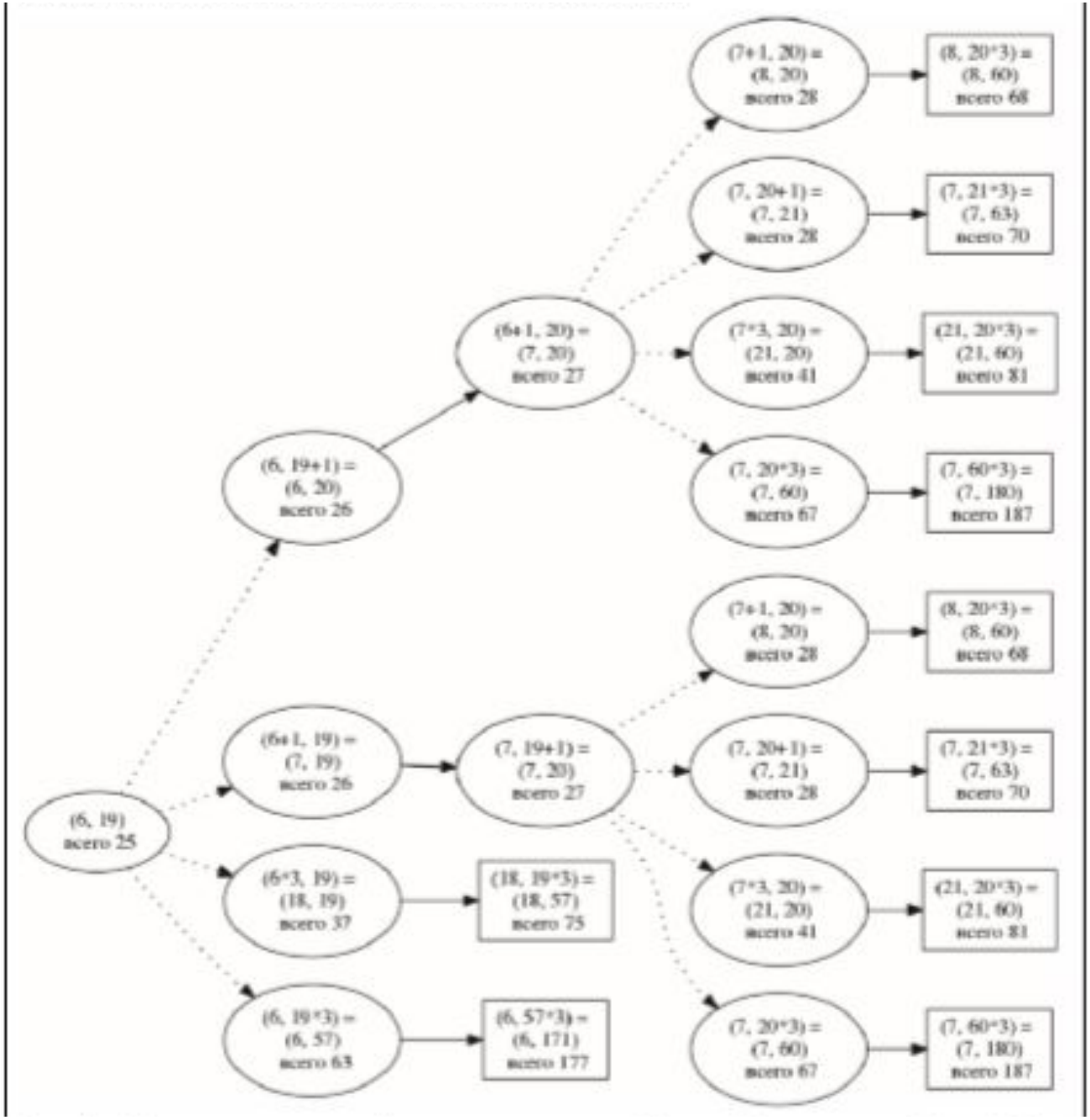
	Положения после очередных ходов			
Исходное положение	1-й ход Пети (разобраны все ходы, указана полученная позиция)	1-й ход Вани (только ход по стратегии, указана полученная позиция)	2-й ход Пети (разобраны все ходы, указана полученная позиция)	2-й ход Вани (только ход по стратегии, указана полученная позиция)
(6, 19) **Всего: 25**	(6, 19+1) = (6, 20) Всего: 26	(6+1, 20) = (7, 20) Всего: 27	(7+1, 20) = (8, 20) Всего: 28	(8, 20*3) = (8, 60) **Всего: 68**
			(7, 20+1) = (7, 21) Всего: 28	(7, 21*3) = (7, 63) **Всего: 70**
			(7*3, 20) = (21, 20) Всего: 41	(21, 20*3) = (21, 60) **Всего: 81**
			(7, 20*3) = (7, 60) Всего: 67	(7, 60*3) = (7, 180) **Всего: 187**
	(6+1, 19) = (7, 19) Всего: 26	(7, 19+1) = (7, 20) Всего: 27	(7+1, 20) = (8, 20) Всего: 28	(8, 20*3) = (8, 60) **Всего: 68**
			(7, 20+1) = (7, 21) Всего: 28	(7, 21*3) = (7, 63) **Всего: 70**
			(7*3, 20) = (21, 20) Всего: 41	(21, 20*3) = (21, 60) **Всего: 81**
			(7, 20*3) = (7, 60) Всего: 67	(7, 60*3) = (7, 180) **Всего: 187**
	(6*3, 19) = (18, 19) Всего: 37	(18, 19*3) = (18, 57) **Всего: 75**		
	(6, 19*3) = (6, 57) Всего: 63	(6, 57*3) = (6, 171) **Всего: 177**		

Примечание для эксперта. Дерево всех партий может быть также изображено в виде ориентированного графа – так, как показано на рисунке, или другим способом. Важно, чтобы множество полных путей в графе

Задание 3

Возможное значение S: 19. После первого хода Пети возможны позиции: (7, 19), (18, 19), (6, 20), (6, 57). В позициях (18, 19) и (6, 57) Ваня может выиграть первым ходом, утроив количество камней во второй куче. Из позиций (7, 19) и (6, 20) Ваня может получить позицию (7, 20). Эта позиция разобрана в п. 2. Игрок, который её получил (теперь это Ваня), выигрывает своим вторым ходом.

В таблице изображено дерево возможных партий (и только их) при описанной стратегии Вани. Заключительные позиции (в них выигрывает


(6, 19)
всего 25
(6, 19+1) =
(6, 20)
всего 26
(6+1, 19) =
(7, 19)
всего 26
(6*3, 19) =
(18, 19)
всего 37
(6, 19*3) =
(6, 57)
всего 63
(6+1, 20) =
(7, 20)
всего 27
(7, 19+1) =
(7, 20)
всего 27
(7+1, 20) =
(8, 20)
всего 28
(7, 20+1) =
(7, 21)
всего 28
(7*3, 20) =
(21, 20)
всего 41
(7, 20*3) =
(7, 60)
всего 67
(8, 20*3) =
(8, 60)
всего 68
(7, 21*3) =
(7, 63)
всего 70
(21, 20*3) =
(21, 60)
всего 81
(7, 60*3) =
(7, 180)
всего 187
(18, 19*3) =
(18, 57)
всего 75
(6, 57*3) =
(6, 171)
всего 177

Допускается наличие не более ... содержательных (не являющимися синтаксическими) ошибок следующих видов:

– допущена ошибка при вводе данных, например не считывается значение N, или числа могут быть считаны, только если будут записаны в одной строке через пробел;

– неверная инициализация или её отсутствие там, где она необходима;

– используется неверный тип данных;

– неверно используется условный оператор, например, else относится не к тому условию;

– отсутствует вывод ответа, или выводится значение не той переменной;

– выход за границу массива;

– неверно расставлены операторные скобки;

– **не более одного раза использована одна переменная (или константа) вместо другой или не более одного раза используется один знак операции вместо другого, или не более одного раза используется одно зарезервированное слово языка программирования вместо другого**.

© все права защищены