Подготовка к ЕГЭ (задания 24, 25, 26, 27)

Выполнила: учитель информатики МБОУ СОШ №34 Лаптева М.А.

24. На обработку поступает натуральное число, не превышающее 10^9. Нужно написать программу, которая выводит на экран сумму цифр числа, не кратных трём. Если в числе нет таких цифр, требуется на экран вывести «NO». Программист написал программу неправильно. Ниже приведена она.

```
var N, digit, sum:longint;
begin
  readln(N);
  sum := 0;
While n>10 do
begin
 digit:=N mod 10;
 if digit mod 3 = 0 then
      sum := sum + digit;
 N:= N div 10;
end;
  if sum > 0 then
    writeln(sum);
  else
    writeln('NO')
end.
```

```
var N, digit,
sum:longint;
begin
  readln(N);
  sum := 0;
While n>10 do
begin
 digit:=N mod 10;
 if digit mod 3 =
0 then
      sum := sum
+ digit;
 N:= N div 10;
end;
  if sum > 0 then
    writeln(sum);
  else
    writeln('NO')
end.
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 357.
2. Приведите пример такого трёхзначного числа, при вводе которого, программа выдаёт 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.

   Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Решение:

1. Программа выводит :”NO”.
2. Программа выдаёт правильный ответ для чисел: 600 или 561. Программа работает неправильно из-за неверного условия цикла и неправильной проверки на некратность 3.
3. В программе есть 2 ошибки:

Строка с ошибкой:	Верное исправление:
While N>10 do	While N>0 do
If digit mod 3 = 0 then	If digit mod 3 <> 0 then

24. На обработку поступает натуральное число, не превышающее 10^9 , и выводится максимальная цифра этого числа. Программист написал программу неправильно. Ниже приведена она.

```
var N:longint;
Digit, max digit: integer;
begin
  readln(N);
 max digit := 0;
While n>=10 do
begin
 digit:=N mod 10;
 if digit < max digit then
   max digit := digit;
 N:= N div 10;
end;
      writeln(max digit);
end.
```

```
var N:longint;
Digit, max digit: integer;
begin
  readln(N);
 max digit := 0;
While n>=10 do
begin
 digit:=N mod 10;
 if digit < max digit then
   max digit := digit;
 N:= N div 10;
end;
      writeln(max digit);
end.
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 762.
2. 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.

   Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Решение:

1. Программа выведет число 0.
2. В программе есть 2 ошибки:
   Первая: неверное условие окончания цикла. Программа не будет рассматривать старшую цифру числа.
   Вторая: неверная операция сравнения при поиске максимума.

Строка с ошибкой:	Верное исправление:
While N>10 do	While N>0 do
If digit <**max digit** then	If digit > **max digit** then

24. На обработку поступает натуральное число, не превышающее 10^9 , и выводится произведение цифр этого числа. Программист написал программу неправильно. Ниже приведена она.

```
var N, product: longint;
 digit: integer;
begin
  readln(N);
  product := 0;
While N > 0 do
begin
 digit:=N mod 10;
 product := product + digit;
N:= N div 10;
end;
      writeln(product);
end.
```

```
var N, product: longint;
 digit: integer;
begin
  readln(N);
  product := 0;
While N > 0 do
begin
 digit:=N mod 10;
 product := product +
digit;
N:= N div 10;
end;
      writeln(product);
end.
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 133.
2. 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.

   Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Решение:

1. Программа выведет число 7.
2. В программе есть 2 ошибки:
   Первая: неверное начальное значение переменной **`product`**.
   Вторая: неверная операция сложения: необходимо найти произведение чисел, а не сумму.

Строка с ошибкой:	Верное исправление:
product:=0	**product:=1**
product:= product + digit	**product:= product * digit**

24. Дано целое положительное число *N*. Необходимо определить наименьшее целое число *K*, для которого выполняется неравенство:
1 + 2 + … + *K* > *N*.

Для решения этой задачи ученик написал программу, но, к сожалению, его программа неправильная. Ниже приведена программа на языке Pascal:

```
var n, k: integer;
begin
    read(n);
    k := 1;
  while n>0 do begin
  n := n- k;
    k := k + 1;
  end;
  writeln(k)
end.
```

Найдите в программе все ошибки (их может быть одна или несколько). Для каждой ошибки выпишите строку, в которой она допущена, и приведите эту же строку в исправленном виде.

Достаточно указать ошибки и способ их исправления для одного языка программирования.

Обратите внимание: Вам нужно исправить приведённую программу, а не написать свою.

Вы можете только заменять ошибочные строки, но не можете удалять строки или добавлять новые. Заменять следует только ошибочные строки: за исправления, внесённые в строки, не содержащие ошибок, баллы будут снижаться.

Решение:

В программе есть 2 ошибки:

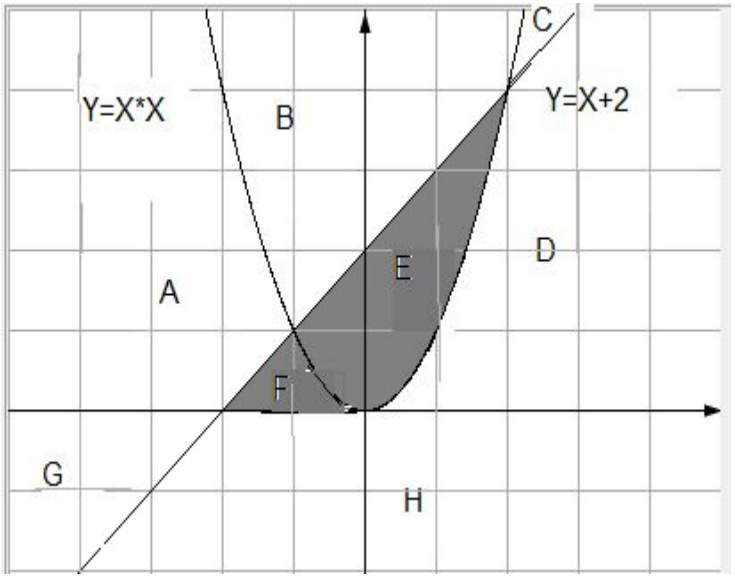
Первая: неверное условие цикла.

Вторая: неверный вывод.

Строка с ошибкой:	Верное исправление:
while *n*>0 do begin	**while *n*>=0 do begin**
writeln(*k*)	**writeln(*k*-1)**

24. Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (x, y – действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы). Программист торопился и написал программу неправильно.

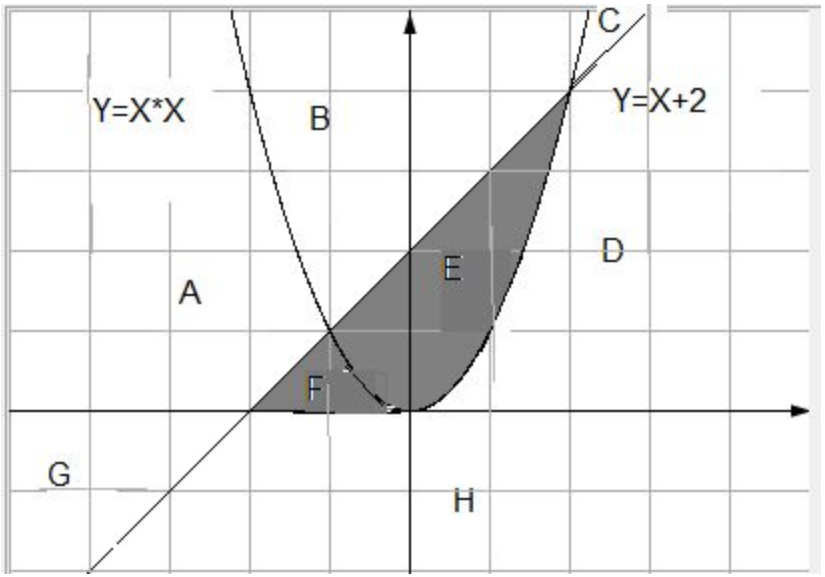
```
var x,y: real;
begin
readln(x,y);
 if y>=0 then
   if y<=x+2 then
     if y>=x*x then
     write('принадлежит')
    else
     write('не принадлежит')
end.
```



1) Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F, G и H). Точки, лежащие на границах областей, отдельно не рассматривать.

   В столбцах условий укажите "да", если условие выполнится, "нет" если условие не выполнится, "—" (прочерк), если условие не будет проверяться, «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области. В столбце "Программа выведет" укажите, что программа выведет на экран. Если программа ничего не выводит, напишите "—" (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите "да" или "нет".

2) Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)




```
var x,y: real;
begin
readln(x,y);
 if y>=0 then
   if y<=x+2 then
      if y>=x*x then
      write('принадлежит')
    else
      write('не принадлежит')
end.
```

Область	y>=0?	y<=x+2	y>=x*x?	вывод	верно?
A					
B					
C					
D					
E					
F					
G					
H					

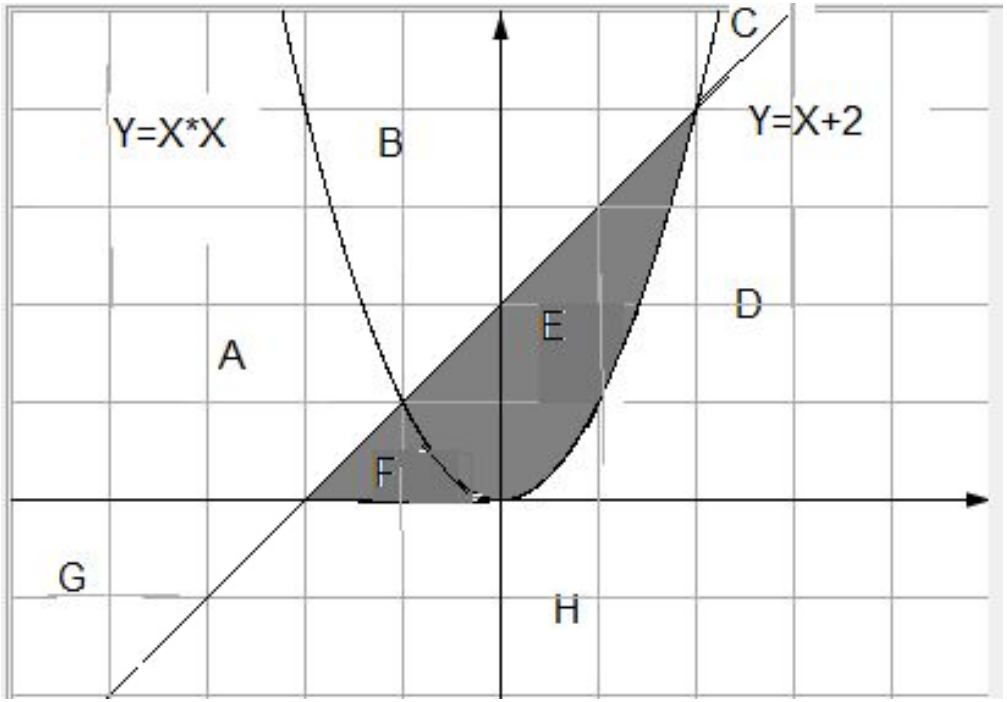
Решение:

Область	y>=0?	y<=x+2	y>=x*x?	вывод	верно?
A	да	нет	-	-	нет
B	да	нет	-	-	нет
C	да	нет	-	-	нет
D	да	да	нет	не принадлежит	да
E	да	да	да	принадлежит	да
F	да	да	нет	не принадлежит	нет
G	нет	-	-	-	нет
H	нет	-	-	-	нет

Условия для заштрихованных областей:

Область E: (y<=x+2) and (y>=x*x)

Область F: ((y<=x+2) and (y<=x*x) and (y>=0)and (x<=0))



Поэтому часть программы после доработки может быть следующего вида:

```
If ((y<=x+2) and (y>=x*x)) or ( (y<=x+2) and
  (y<x*x) and (y>=0) and (x<=0)) then
  write('принадлежит')
 else
     write('не принадлежит');
```

Или после упрощения логического выражения:

```
If (y<=x+2) and ((y>=x*x) or ( (y<x*x) and (y>=0)
  and (x<=0))) then
  write('принадлежит')
 else
  write('не принадлежит');
```

25. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от –10 000 до 10 000 включительно. Опишите на естественном языке или на одном из языков программирования алгоритм, позволяющий найти и вывести количество пар элементов массива, в которых хотя бы одно число не делится на 7. Под парой подразумевается два подряд идущих элемента массива.

Исходные данные объявлены так, как показано ниже.

Запрещается использовать переменные, не описанные ниже, но использовать все описанные переменные не обязательно.

```
const N = 30;
var a: array [1..N] of integer;
    i, j, k: integer;
begin
  for i := 1 to N do
    readln(a[i]);
  ...
End.
```

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

Решение:

```
 k:= 0;
for i: =1 to N-1 do
 if (a(i) mod 7 < > 0) or (a(i+1) mod 7 < > 0) then
   inc ( k ) ;
writeln (k);
```

25. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 10 000 включительно. Опишите на естественном языке или на одном из языков программирования алгоритм, позволяющий найти и вывести максимальное значение среди трёхзначных элементов массива, не делящихся на 20. Если в исходном массиве нет трёхзначного элемента, не кратного 20, то вывести сообщение «не найдено».

Исходные данные объявлены так, как показано ниже.

Запрещается использовать переменные, не описанные ниже, но использовать все описанные переменные не обязательно.

```
const
   N = 30;
var a: array [1..N] of integer;
    i, j, max: integer;
begin
  for i := 1 to N do
    readln(a[i]);
  ...
End.
```

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

Решение:

```
 max:= 99;
for i: =1 to N do
 if (a(i)  > = 100) and (a(i) < = 999) and (a(i) mod 20 <> 0) and (a(i) > max)
then
   max := a (i) ;
if max > 99 then writeln (max) else writeln (“ не найдено”) ;
```

25. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от -1000 до 1000 включительно. Опишите на естественном языке или на одном из языков программирования алгоритм, позволяющий найти и вывести максимальное значение среди отрицательных элементов массива, не оканчивающихся на 3. Если в исходном массиве нет элемента, значение которого отрицательно и не оканчивается на цифру 3, то вывести сообщение «не найдено».

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но использовать все описанные переменные не обязательно.

```
const
   N = 30;
var a: array [1..N] of integer;
    i, j, max: integer;
begin
  for i := 1 to N do
    readln(a[i]);
  ...
End.
```

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

Решение:

```
 max:= -1001;
for i: =1 to N do
 if (a(i) < 0) and ( (-1 * a(i)) mod 10 <> 3) and (a(i) > max) then
   max := a (i) ;
if max > -1001 then writeln (max) else writeln (“ не найдено”) ;
```

25. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 10000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет найти и вывести произведение элементов массива, которые имеют двузначное значение и не оканчиваются на 2. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого является двузначным числом, и при этом его последняя цифра не равна двум.

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

```
Const
   N=30;
Var
  a: array [1..N] of integer;
   i, j, p: longint;
begin
  for i:=1 to N do
      readln(a[i]);
 ...
end.
```

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, *Borland Pascal 7.0*) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Решение:

```
P:=1;
For i:=1 to n do
   if (a[i]>9) and (a[i]<100) and (a[i] mod 10 <> 2)
   then
    P:=P*a[i];
Writeln (P);
```

26. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **1** или **3** камня или увеличить количество камней в куче в **2 раза**.

Например имея кучу из 12 камней, за один ход можно получить кучу из 13, 15 или 24 камней. У каждого игрока, чтобы делать ходы есть неограниченное количество камней.

Игра завершается, когда количество камней в куче становится не менее 40.

Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 40 и более камней.

В начальный момент в куче было S камней, 1 <= S <= 39.

Будем говорить, что игрок **имеет выигрышную стратегию**, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать какой ход он может сделать в любой ситуации, которая может ему встретиться при различной игре противника.

Выполните следующие задания, во всех случаях обосновывая ответ.

1. а. Укажите все значения числа S, при которых Петя может выиграть в один ход. Обоснуйте, что найдены все нужные значения S и укажите выигрывающий ход для каждого значения.

   б. Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

2. Укажите два таких значения S, при которых у Пети есть выигрышная стратегия, причём Петя не может выиграть за один ход и Петя может выиграть своим вторым ходом независимо от того как будет ходить Ваня. Для каждого значения S опишите выигрышную стратегию Пети.

3. Укажите значение S, при котором:

   а. У Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторы ходом при любой игре Пети.

   в. 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).

Решение:

Задание 1.

а. Петя может выиграть в один ход, если S=20,…39. Во всех этих случаях достаточно удвоить количество камней, после чего количество станет больше 39, и игра закончится.

б. Ваня может выиграть первым ходом (при любой игре Пети), если S=19. Тогда после первого хода Пети в куче будет 20, 22 и 38 камней. Далее Ваня удваивает количество камней и выигрывает в один ход.

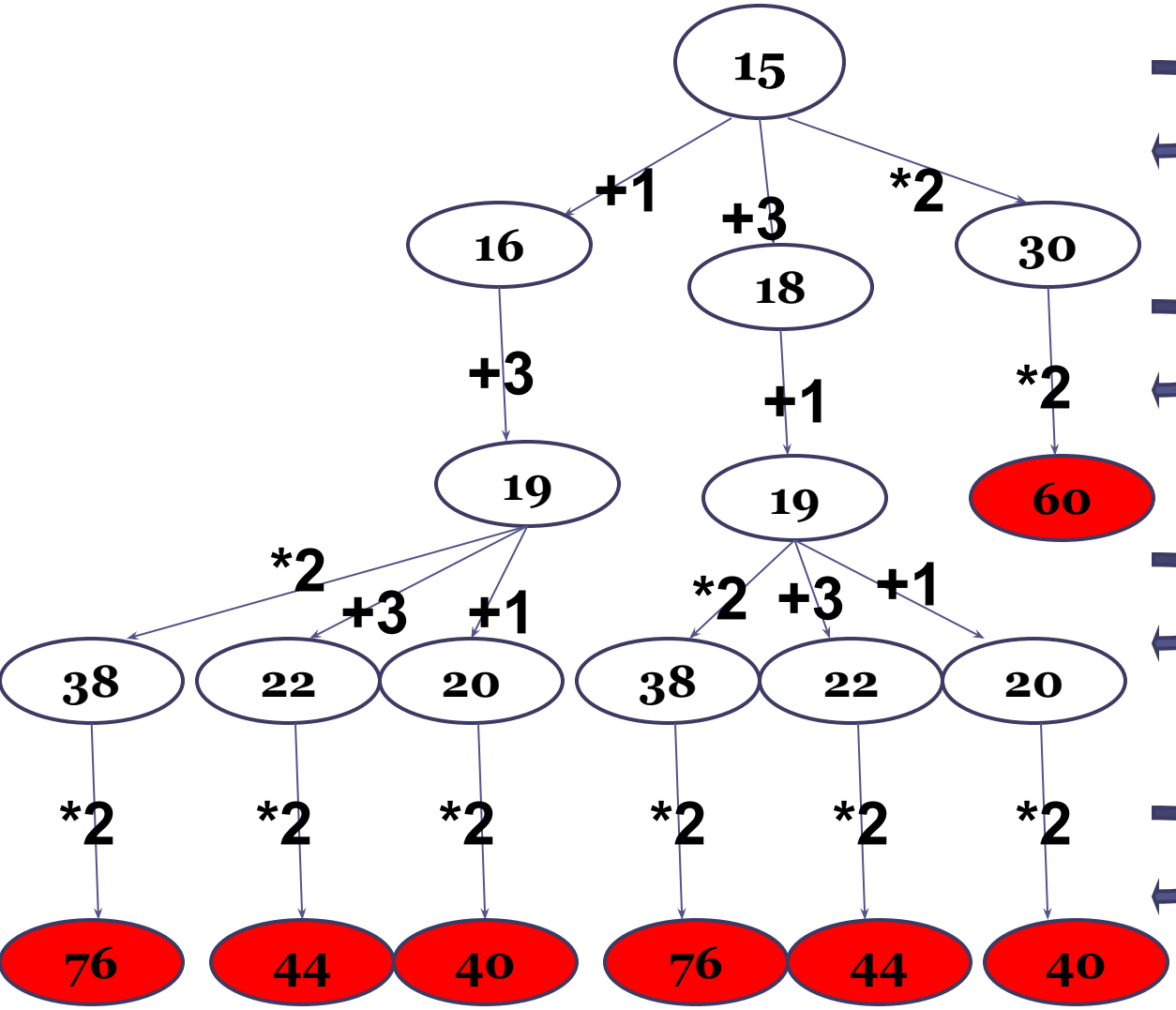
Задание 2.

При S=16 или S=18 у Пети есть выигрышная стратегия, позволяющая ему выиграть своим вторым ходом, не выигрывая первым ходом. Он может получить кучу из 19 камней, добавив в кучу 1 камень (при S=18) или 3 камня (при S=16). После этого хода Петя попадает в ситуацию, разобранную в п. 1б для Вани, то есть у игрока, делающего следующий ход (у Вани), нет хода, сразу приводящего его к выигрышу, а у Пети выигрышный ход «удвоить количество камней» есть независимо от того, какой ход сделал Ваня.

Задание 3.

При S=15 у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом. После первого хода Пети в куче будет 16, 18 или 20 камней. Если в куче станет 30 камней, то Ваня удвоит количество камней и выиграет своим первым ходом.

Если после первого хода Пети в куче оказалось 16 или 18 камней Ваня попадает в ситуацию, разобранную в п. 2 для Пети и у него есть выигрышная стратегия, позволяющая ему выиграть своим вторым ходом.



1-й ход Пети

1-й ход Вани
(выигрышные ходы)

2-й ход Пети
(все ходы)

2-й ход Вани
(выигрышные ходы)

26. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один камень** или увеличить количество камней в куче в **два раза**. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 34. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 34 или больше камней.

В начальный момент в куче было S камней, $1 \le S \le 33$.

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.

Выполните следующие задания. Во всех случаях обосновывайте свой ответ.

1. а) Укажите все такие значения числа S, при которых Петя может выиграть в один ход. Обоснуйте, что найдены все нужные значения S, и укажите выигрывающий ход для каждого указанного значения S.

Решение:

Последним ходом может быть «+1» или «*2». Выиграть последним ходом «+1» можно, если S = 33.

Ходом «*2» можно выиграть из любой позиции при S >= 17 (или S>16) (сюда входит и 33!).

Ответ для 1а. Петя может выиграть за один ход при любом S > 16. Он должен увеличить вдвое число камней, при этом в куче всегда получится не менее 34 камней.

1. б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

Решение:

Для ответа на этот вопрос нужно найти позицию, из которой все возможные ходы ведут к выигрышу за 1 ход, то есть к позициям, найденным в пункте 1а.

Ответ для 1б: При S = 16 Петя не может выиграть в один ход, потому что при его ходе «+1» число камней в куче становится равно 17 (меньше 34), а при ходе «*2» число камней в куче становится равно 32 (также меньше 34). Других возможных ходов у Пети нет.

Из любой позиции после одного хода Пети (это может быть 17 или 32), Ваня может выиграть своим первых ходом, удвоив количество камней в куче.

2. Укажите два таких значения S, при которых у Пети есть выигрышная стратегия, причём

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

Для каждого указанного значения S опишите выигрышную стратегию Пети.

Решение:

Пете, для того, чтобы гарантированно выиграть на втором ходу, нужно из начальной позиции перевести игру в одну из проигрышных позиций, например, S = 16 (см п.1б). Петя может перевести игру в эту позицию из позиций S = 15 (ходом «+1») и S = 8 (ходом «*2»).

Ответ для п.2: из позиций S = 15 и S = 8 Петя не может выиграть в один ход, но Петя может выиграть своим вторым ходом, независимо от того, как будет ходить Ваня. При S = 15 ходом «+1» Пете нужно перевести игру в позицию S = 16, которая является проигрышной (см. ответ на вопрос 1б). При S = 8 Петя переводит игру в ту же позицию ходом «*2».

3. Укажите значение S, при котором:
– у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, и
– у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Для указанного значения S опишите выигрышную стратегию Вани.

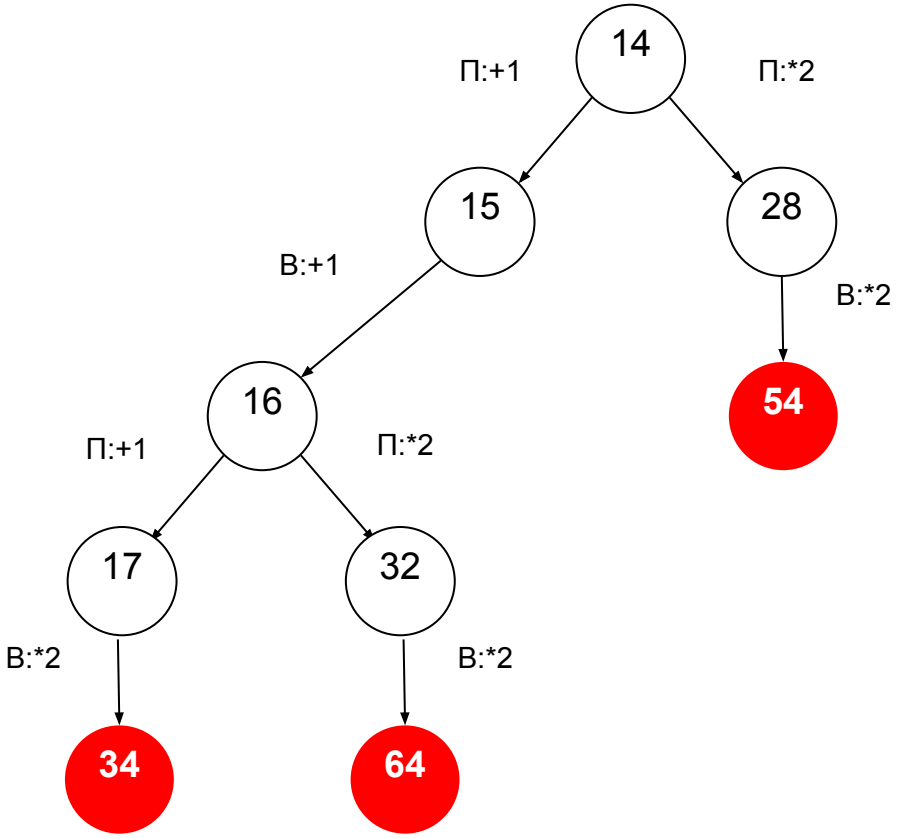
Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы). На рёбрах дерева указывайте, кто делает ход, в узлах – количество камней в куче.

Решение:

Нужно найти такую позицию, из которой оба возможных хода Пети ведут к проигрышу, т.е. в Петя не должен попасть в позицию, из которых возможны и выигрыш в 1 ход, и выигрыш в 2 хода. Например, это позиция S = 14, из которой можно «попасть» только в S = 15 («выигрыш в два хода») и S = 28 («выигрыш в один ход»).

Ответ для п.3: В позиции S = 14 у Вани есть выигрышная стратегия, которая позволяет ему выиграть первым или вторым ходом. Если Петя выбирает ход «+1», в куче становится 15 камней и Ваня выигрывает на 2-м ходу (см. ответ на вопрос 2). Если Петя выбирает ход «*2», Ваня выигрывает первым ходом, удвоив число камней в куче.

Остается нарисовать дерево возможных вариантов игры из позиции S = 14.



Обратите внимание, что на каждом шаге мы рассматриваем все возможные ходы Пети и только один лучший ход Вани.

Все ходы Вани мы не рассматриваем, потому что мы хотим доказать, что **у Вани** есть выигрышная стратегия – ему достаточно одного хода, после которого он выиграет.

В то же время нужно рассмотреть **все возможные ответы Пети**, чтобы доказать, что у него нет шансов на выигрыш при правильной игре Вани. В этом суть теории игр – добиться лучшего результата в худшем случае, то есть при безошибочной игре соперника.

26. Два участника играют в следующую игру. На координатной плоскости стоит фишка. Игроки играют по очереди. В начале игры фишка находится в точке с координатами (3; 2). Ход состоит в том, что игрок перемещает фишку из точки с координатами (x;y) в одну из трёх точек: в точку с координатами (x+1;y), или в точку с координатами (x+2;y), в точку с координатами (x;y+2).
Выигрывает игрок, после хода которого расстояние от фишки до точки с координатами (0; 0) не меньше 8 единиц.
Кто выигрывает при правильной игре? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Решение:

Выигрывает первый игрок. Его первый ход – в точку (5, 2). Для доказательства представлено неполное дерево игры в виде таблицы: для первого игрока представлены выигрышные ходы, для второго – все варианты ходов, возможные после очередного хода первого игрока.

Старт	**1-й ход**	**2-й ход**	**3-й ход**
	1 игрок	**2 игрок**	**1 игрок**
(3;2)	**(5;2)**	(6;2)	**(8;2)**
		(7;2)	**(8;2)**
		(5;4)	**(7;4)**

26. Два участника играют в игру «три кучки». В их распоряжении три кучки камней. Каждым ходом игрок может взять от 1 до 3 камней, но только из одной из трёх куч. Проигрывает тот, кто взял последний камень. Укажите, у какого из игроков есть выигрышная стратегия, и опишите её, если известно, что изначально в первой куче было 3 камня, во второй тоже 3, а в третьей – 2..

Примечание. Оба игрока стремятся выиграть и поэтому не делают заведомо проигрышные ходы. Говорят, что у игрока есть выигрышная стратегия, если он может играть так, чтобы победить при любых действиях оппонента.

Решение:

Первым ходом первый игрок должен взять из первой или второй кучи 2 камня (получится 1 3 2) или из третьей 2 (получится 3 3 0). Дальнейшее поведение первого игрока в зависимости действий второго описаны в таблице:

Ситуация после хода I игрока	**II игрок**	**I игрок**	**II игрок**	**I игрок**
1 3 2	0 3 2	0 2 2	0 1 2	1 0 0
	1 2 2		0 0 2	
	1 3 1	1 1 1	0 1 1	
	1 1 2			
	1 3 0			
	1 0 2			

Ситуация после хода I игрока	**II игрок**	**I игрок**	**II игрок**	**I игрок**
3 3 0	2 3 0	2 2 0	2 1 0	1 0 0
			2 0 0	
	1 3 0			
	0 3 0			

27. Пусть дана последовательность A целых чисел, пронумерованных от 1 до N. Будем называть невозрастающей подпоследовательностью подряд идущие элементы последовательности, такие что $A_I >= A_{I+1} >= \ldots >= A_{K+1} >= A_K$ (K>=I>0). Длиной последовательности будем называть количество входящих в него элементов. Любой отдельно взятый элемент представляет собой невозрастающую подпоследовательность длины 1. От цифровых датчиков в ПК поступает информация о характеристиках физического процесса. Результатом
каждого измерения является целое число. Предлагается написать эффективную программу, которая будет искать максимальную длину невозрастающей последовательности.
Следует учитывать, что количество измерений может быть очень велико.
Перед текстом программы кратко опишите используемый ВАМИ алгоритм решения задачи. На вход программе в первой строке подаётся общее количество N значений измерений. В каждой из последующих N строк записано целое число. Гарантируется, что N>= 1, т.е. всегда имеется хотя бы одно значение измерений.
Пример входных данных:
5
-1000
0
-300
2
2000
Результатом работы программы является целое число – максимальная длина невозрастающей последовательности. Ответ:2

Решение:

Программа читает значения измерений, обновляя при необходимости длину искомой последовательности.

Программа на языке Pascal:

```
var N, cnt, I, t, prev, max: integer
begin
max:=1;
cnt:=1;
ReadIn (N);  {Считываем количество измерений}
 for I:=1 to N do
begin
ReadIn (t);  {Считываем очередное значение}
If i=1 then prev:= t
   else
        If prev >=t then cnt:= cnt+1
   else
     begin
        If cnt > max then
        max:= cnt; {Обновление максимальной длины}
        cnt:=1;
     end;
   prev:=t;
end;
If cnt > max then
        max:= cnt; {Обработка хвоста последовательности}
writeln (max);
```

27. На вход программе подается зашифрованный текст заклинания, состоящего не более, чем из 200 символов. Этот текст представляет собой последовательность слов (т.е. непрерывных последовательностей английских букв длиной не более 20 символов), разделенных произвольным количеством пробелов и знаков препинания («,», «.», «:», «-»), которая заканчивается точкой. Символы входной строки после точки, если они есть, к заклинанию не относятся и поэтому игнорируются. Заклинание было зашифровано Гарри Поттером следующим образом. Сначала Гарри определил количество букв в самом длинном слове, обозначив полученное число K. Затем он заменил каждую английскую букву в заклинании на букву, стоящую в алфавите на K букв далее (алфавит считается циклическим, то есть, после буквы Z стоит буква A), оставив другие символы неизменными. Строчные буквы при этом остались строчными, а прописные – прописными.

Требуется написать программу как можно более эффективную программу, которая будет выводить на экран текст расшифрованного заклинания заклинания. Например, если исходный текст был таким:

Ce Ud Fd Gde Ud.

то результат расшифровки должен быть следующий:

Zb Ra Ca Dab Ra.

Решение:

Из условия следует, что задача решается в два этапа:

1) прочитать символы до точки и определить длину самого длинного слова из латинских букв (обозначим ее maxLen);

2) сделать «сдвиг» кодов латинских букв на maxLen влево.

Простое посимвольное чтение строки **s** до первой встреченной точки выглядит так (здесь **c** – переменная типа **char**):

```
s := ''; { пустая строка }
repeat
  read(c);    { прочитали символ }
  s := s + c; { добавили в конец строки }
until c = '.';
```

При этом нам нужно еще определить длину самого длинного слова с учетом того, что между словами может быть сколько угодно символов-разделителей (разных!).

Введем переменную **len**, которая будет определять длину текущего (очередного, вводимого в данный момент) слова.

Как определить, что прочитанный символ – латинская буква? Можно использовать условный оператор со сложным условием:

```
if (('a' <= c) and (c <= 'z')) or
   (('A' <= c) and (c <= 'Z')) then ...
```

Или это можно сделать с помощью оператора in, который проверяет, входит ли элемент во множество:

```
if c in ['a'..'z', 'A'..'Z'] then ...
```

Здесь множество в квадратных скобках содержит два интервала: от 'a' до 'z' и от 'A' до 'Z'.

Если очередной прочитанный символ – латинская буква, нужно увеличить **len** на единицу (слово продолжается). Если же это не латинская буква, то слово закончилось, так как встречен символ-разделитель .

Полученное значение переменной **len** нужно сравнить с максимальной длиной и, если прочитанное слово длиннее всех предыдущих, записать его длину в **maxLen**. Таким образом, цикл ввода выглядит так:

```
s := '';
maxLen := 0;
len := 0;
repeat
    read(c);
    s := s + c;
    if c in['a'..'z','A'..'Z'] then
      len := len + 1
    else
       begin
          if len > maxLen then
             maxLen := len;
          len := 0;
       end;
until c = '.';
```

Теперь нужно в цикле пройти всю прочитанную строку и «сдвинуть» каждый символ (точнее, его код) вправо на **maxLen**:

```
for i:=1 to Length(s) do
  if s[i] in ['a'..'z','A'..'Z'] then
    begin
      code := Ord(s[i]);  { старый код }
      newcode := code - maxLen;
      { новый код }
      s[i] := Chr(newcode);
    end;
```

Однако такое решение не учитывает цикличность: например, при сдвиге буквы **'A'** на 2 символа влево мы не получим **'Y'**. Поэтому после изменения кода нужно проверить, не вышел ли он за допустимые границы (диапазона латинских букв), а если вышел, то добавить к полученному коду 26 (число латинских букв), что обеспечит циклический сдвиг:

```
newcode := code - maxLen;
{ новый код }
{ цикличность }
if s[i] in ['a'..'z'] then
  if newcode < Ord('a') then
    newcode := newcode + 26;
if s[i] in ['A'..'Z'] then
  if newcode < Ord('A') then
    newcode := newcode + 26;
```

Программа на языке «Pascal»:

```
var c: char;
    s: string;
    len, maxLen, code, i, newcode : integer;
begin
  s := '';
  maxLen := 0;
  len := 0;
  { чтение данных }
  repeat
    read(c);
    s := s + c;
    if c in['a'..'z','A'..'Z'] then
      len := len + 1
    else
      begin
        if len > maxLen then
          maxLen := len;
        len := 0;
      end;
  until c = '.';
```

```
{ сдвиг кодов на maxLen влево }
 for i:=1 to Length(s) do
   if s[i] in ['a'..'z','A'..'Z'] then
     begin
       code := Ord(s[i]);  { старый код }
       newcode := code - maxLen; { новый код }
       { цикличность }
       if s[i] in ['a'..'z'] then
         if newcode < Ord('a') then
            newcode := newcode + 26;
        if s[i] in ['A'..'Z'] then
          if newcode < Ord('A')
             then newcode := newcode + 26;
       { запись нового кода }
       s[i] := Chr(newcode);
     end;
  writeln(s);
end.
```

Список используемых источников:

- Сайт Константина Полякова “Материалы для подготовки к ЕГЭ ”httpСайт Константина Полякова “Материалы для подготовки к ЕГЭ ”http://Сайт Константина Полякова “Материалы для подготовки к ЕГЭ ”http://kpolyakovСайт Константина Полякова “Материалы для подготовки к ЕГЭ ”http://kpolyakov.Сайт Константина Полякова “Материалы для подготовки к ЕГЭ ”http://kpolyakov.narodСайт Константина Полякова “Материалы для подготовки к ЕГЭ ”http://kpolyakov.narod.Сайт Константина Полякова “Материалы для подготовки к ЕГЭ ”http://kpolyakov.narod.ru
- ЕГЭ. Информатика и ИКТ: типовые экзаменационные вопросы Е31 10 вариантов / С.С. Крылов, Т.Е. Чуркина. – М.: Издательство

Успехов Вам и Вашим ученикам!!!