Министерство науки и высшего образования РФ
Соликамский государственный педагогический институт (филиал)
федерального государственного автономного образовательного
учреждения высшего образования
«Пермский государственный национальный исследовательский университет»

Утверждено на заседании
Ученого совета СГПИ филиала ПГНИУ
Протокол № 10
от «14» октября 2021

Утверждено на заседании
кафедры математических и естественнонаучных
дисциплин СГПИ филиала ПГНИУ
Протокол № 2
от «05» октября 2021

ВСТУПИТЕЛЬНЫЙ ЭКЗАМЕН ПО ОСНОВАМ ПРОГРАММИРОВАНИЯ

1. ОБЩИЕ УКАЗАНИЯ

Цель проведения вступительного экзамена по основам программирования – оценить уровень теоретической и практической подготовки абитуриентов в разработке алгоритмов и программ на процедурно-ориентированном языке программирования при решении задач.

Максимальное количество баллов, подтверждающих успешное прохождение вступительного испытания - **100 баллов**.

Минимальное количество баллов, подтверждающих успешное прохождение вступительного испытания - **40 баллов**.

Задачи проведения вступительного экзамена по основам программирования:

– выявить необходимые знания основ алгоритмизации, методов разработки программ, основных элементов и принципов программирования на процедурно-ориентированном языке программирования;

– оценить навыки разработки алгоритмов и программ на одном из процедурных языков программирования (например, Паскаль или Си).

В результате прохождения вступительного экзамена абитуриенты должны продемонстрировать *знания:*

– понятия алгоритма, свойств алгоритмов, общих принципов построения алгоритмов, способов описания алгоритмов, типов алгоритмов;

– принципов построения эффективных алгоритмов;

– методов разработки программ, основных элементов языка программирования, операторов, функций и операций, управляющих структур, структур данных, файлов;

– принципов программирования на одном из алгоритмических языков высокого уровня (как, например, Pascal, или C, или C++, или Java, или т. п.);

умения:

– разрабатывать алгоритм решения задачи;

– разрабатывать программы, проводить их отладку, тестирование и верификацию.

2. ТРЕБОВАНИЯ К ВСТУПИТЕЛЬНОМУ ЭКЗАМЕНУ

2.1 Содержание программы вступительного экзамена

ТЕМА 1. ОБЩИЕ СВЕДЕНИЯ ОБ АЛГОРИТМАХ

Алгоритм и его свойства. Способы описания алгоритмов. Стандартизация графического представления алгоритмов. Методы разработки и анализа алгоритмов. Псевдокоды. Проблемы создания алгоритмов. Проблема универсального языка программирования. Проблема универсальной вычислительной машины. Машина Тьюринга. Написание программ на машине Тьюринга.

ТЕМА 2. ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА

Основные понятия языка. Структура программы. Простые типы данных. Операции и их приоритет. Выражения. Основные операторы. Основные возможности организации ввода/вывода. Стандартные потоки ввода/вывода.

ТЕМА 3. ТИПЫ ДАННЫХ

Представление чисел в памяти компьютера. Позиционные системы счисления. Перевод чисел из одной системы счисления в другую. Массивы. Работа с массивами. Строки. Работа со строками. Структуры данных различного типа. Работа со структурами. Адреса и указатели. Основные возможности работы с динамической памятью. Задачи поиска и сортировки. Бинарный и интерполяционный поиск. Сортировка методом «пузырька», вставками, подсчетом, слиянием.

ТЕМА 4. ПОДПРОГРАММЫ

Модульность в программировании. Понятие и структура подпрограммы. Описание подпрограмм в языках высокого уровня (процедуры, функции). Организация вызова подпрограммы. Типы параметров подпрограммы (функции), локальные и глобальные переменные. Организация многофункциональных программ. Внешние модули. Рекурсивные алгоритмы и функции.

ТЕМА 5. ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

Организация динамических структур данных. Списки. Стеки. Очереди. Организация данных в виде древовидных динамических структур. Двоичные деревья. Деревья бинарного поиска и основные операции для работы с ними: поиск, вставка, удаление. Принципы хеширования. Хеширование с открытой и закрытой адресацией.

ТЕМА 6. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ

Специфические операторы языка программирования. Специфические типы данных. Специфические возможности языка программирования. Базовые принципы объектно-ориентированного программирования: инкапсуляция, наследование, полиморфизм. Понятие объекта, класса, метода, свойства. Конструкторы, деструкторы.

КРИТЕРИИ ОЦЕНКИ

Продолжительность экзамена – 60 минут.
Количество заданий – 20.
Экзамен проводится в тестовой форме.
Максимальное количество баллов – 100 (табл. 1).

Таблица 1

Максимальное количество баллов за выполнение каждого задания

№ задания	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Балл	3	3	3	4	4	4	4	5	5	5	5	5	5	6	6	6	6	7	7	7

Задание считается выполненным, если ответ удовлетворяет следующим критериям:

– правильно выбран способ решения задачи;

– даны точные определения необходимых терминов;

– временная сложность программы удовлетворяет требованиям условия задачи;

– получен правильный ответ.

Примеры экзаменационных заданий (с решениями)

Задание 1

У исполнителя, который работает с положительными однобайтовыми двоичными числами, две команды, которым присвоены номера:

1. сдвинь влево

2. вычти 1

Выполняя первую из них, исполнитель сдвигает число на один двоичный разряд влево, причём на место освободившегося бита ставится 0. Выполняя вторую команду исполнитель вычитает из числа 1. Исполнитель начал вычисления с числа 91 и выполнил цепочку команд 112112. Запишите результат в десятичной системе.

Решение

Если в старшем разряде двоичного числа нет единицы, то команда 1 удваивает число, если единица есть (т. е. десятичное число не меньше 128), то выводится остаток от деления удвоенного числа на 256. Таким образом, получим следующее:

1: 91 => 182,
1: 182 => 108 (остаток от 364 / 256),
2: 108 => 107,
1: 107 => 214,
1: 214 => 172 (остаток от 428 / 256),
2: 172 => 171.

Ответ: 171.

Задание 2

Определите, что будет напечатано в результате работы следующего фрагмента программы:

Паскаль	Алгоритмический язык

var k, s: integer;
begin
  s:=0;
  k:=1;
  while s < 66 do begin
  k:=k+3;
  s:=s+k;
  end;
  write(k);
end.
	алг
	нач
	  цел k, s
	  s := 0
	  k := 1
	  нц пока s < 66
	  k := k + 3
	  s := s + k
	  кц
	  вывод k
	кон

Си++

```
#include <iostream>
using namespace std;
int main() {
  int s, k;
  s = 0, k = 1;
  while (s < 66) {
    k = k + 3;
    s = s + k;
  }
  cout << k << endl;
  return 0;
}
```

Решение

Составим таблицу, в которую занесём переменные *s* и *k*. Будем заполнять эту таблицу до тех пор пока выполняется условие цикла:

s	0	4	11	21	34	50	69
k	1	4	7	10	13	16	19

Цикл прервётся, когда переменная *s* станет равна 69. Переменная *k* при этом будет равна 19.

Задание 3

Исполнитель Редактор получает на вход строку цифр и преобразовывает её. Редактор может выполнять две команды, в обеих командах *v* и *w* обозначают цепочки цифр.

А) **заменить** (*v, w*).

Эта команда заменяет в строке первое слева вхождение цепочки *v* на цепочку *w*. Например, выполнение команды

заменить (555, 63)

преобразует строку 12555550 в строку 1263550.

Если в строке нет вхождений цепочки *v*, то выполнение команды **заменить** (*v, w*) не меняет эту строку.

Б) **нашлось** (*v*).

Эта команда проверяет, встречается ли цепочка *v* в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение «истина», в противном случае возвращает значение «ложь». Строка исполнителя при этом не изменяется.

Цикл
ПОКА *условие*
  последовательность команд
КОНЕЦ ПОКА
выполняется, пока условие истинно.
В конструкции
ЕСЛИ *условие*
  ТО *команда1*
  ИНАЧЕ *команда2*
КОНЕЦ ЕСЛИ
выполняется *команда1* (если условие истинно) или *команда2* (если условие ложно).

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 1000 идущих подряд цифр 9? В ответе запишите полученную строку.

```
НАЧАЛО
ПОКА нашлось (999) ИЛИ нашлось (888)
 ЕСЛИ нашлось (888)
  ТО заменить (888, 9)
  ИНАЧЕ заменить (999, 8)
 КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Решение.

По ходу работы программы строка будет меняться так: 9999... → 89999... → 889999... → 8889999... → 9999...

При этом за каждые 4 итерации цикла будет убираться 3 + 3 + 3 - 1 = 8 девяток.

Тогда после 124 раз по 4 итерации, то есть после 496 итераций, из строки будет убрано 992 девятки и она примет вид 99999999. После чего цикл отработает ещё пару раз: 99999999 → 899999 → 8899.

Задание 4

Ниже на языках программирования записан рекурсивный алгоритм *F*.

Паскаль	**Алгоритмический язык**

procedure F(n: integer);
begin
  if n > 0 then
  begin
  F(n div 4);
  write(n);
  F(n - 1);
  end
end;
	алг F(цел n)
	нач
	  если n > 0 то
	  F(div(n, 4))
	  вывод n
	  F(n - 1)
	  все
	кон

Си++	

void F(int n){
  if (n > 0){
  F(n / 4);
  std::cout << n;
  F(n - 1);
  }
}	

В качестве ответа укажите последовательность цифр, которая будет напечатана на экране в результате вызова F(5).

Решение

Моделируем работу алгоритма.

```
F(5):F(1):F(0)
          1
          F(0)
     5
     F(4):F(1):1
           4
           F(3):F(0)
                 3
                 F(2):F(0)
                       2
                       F(1)
                       1
```

Таким образом, при выполнении вызова F(5) будут напечатаны числа 1514321.

Ответ: 1514321.

Задание 5

В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от −10 000 до 10 000 включительно. Определите и запишите в ответе сначала количество пар элементов последовательности, в которых хотя бы одно число делится на 3, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности. Например, для последовательности из пяти элементов: 6; 2; 9; –3; 6 — ответ: 4 11.

Решение

Будем последовательно считывать числа из файла. Для каждой пары (двух подряд идущих элементов) будем проверять, делится ли хотя бы одно число из пары на 3. При успешном выполнении условия будем увеличивать значения счётчика count и проверять, больше ли сумма элементов пары текущей максимальной суммы. Если сумма элементов пары больше текущей максимальной суммы, будем обновлять значение переменной maxsum.

Решение задачи на языке Pascal

```
var
x, y, count: longint;
maxsum: longint;
f: text;
begin
assign(f,'C:\17.txt');
reset(f);
readln(f, x);
maxsum := 0;
count := 0;
while not eof(f) do begin
readln(f, y);
if (x mod 3 = 0) or (y mod 3 = 0) then begin
count := count + 1;
if x + y > maxsum then maxsum := x + y;
end;
x := y;
end;
writeln(count, ' ', maxsum);
end.
```

В результате работы данного алгоритма при вводе данных из файла ответ — 2802 1990.

Задание 6

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 один камень**, увеличить **количество камней в первой куче в два раза** или **увеличить количество камней во второй куче в три раза**. Например, пусть в одной куче 6 камней, а в другой 9 камней; такую позицию мы будем обозначать (6, 9). За один ход из позиции (6, 9) можно получить любую из четырёх позиций: (7, 9), (12, 9), (6, 10), (6, 27). Чтобы делать ходы, у каждого игрока есть неограниченное количество камней.

Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 69. Победителем считается игрок, сделавший по-

следний ход, то есть первым получивший позицию, в которой в кучах будет 69 или больше камней.

В начальный момент в первой куче было 10 камней, во второй куче — S камней, $1 \le S \le 58$.

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 ней игрока, которые не являются для него безусловно выигрышными, т.е не гарантирующие выигрыш независимо от игры противника.

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

Решение.

Такая ситуация возможна при $S = 7$. Если Петя утроит вторую кучу, получится позиция (10, 21), из которой Ваня может получить позицию (10, 63) и выиграть. При $S < 7$ никакой первый ход Пети не создаст ситуацию, в которой Ваня может сразу выиграть.

Ответ: 7.

Задание 7

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 один камень**, увеличить **количество камней в первой куче в два раза** или **увеличить количество камней во второй куче в три раза**. Например, пусть в одной куче 6 камней, а в другой 9 камней; такую позицию мы будем обозначать (6, 9). За один ход из позиции (6, 9) можно получить любую из четырёх позиций: (7, 9), (12, 9), (6, 10), (6, 27). Чтобы делать ходы, у каждого игрока есть неограниченное количество камней.

Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 69. Победителем считается игрок, сделавший последний ход, то есть первым получивший позицию, в которой в кучах будет 69 или больше камней.

В начальный момент в первой куче было 10 камней, во второй куче — S камней, $1 \le S \le 58$.

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 ней игрока, которые не являются для него безусловно выигрышными, т.е не гарантирующие выигрыш независимо от игры противника.

Найдите два таких значения S, при которых 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.

Найденные значения запишите в ответе в порядке возрастания без разделительных знаков.

Решение.

Возможные значения S: 16, 19. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако при $S = 16$ Петя может получить позицию (20, 16), а при $S = 19$ — позицию (11, 19).

В первом случае после хода Вани возникнет одна из позиций (21, 16), (40, 16), (20, 17), (20, 48), во втором случае — одна из позиций (12, 19), (22, 19), (11, 20), (11, 57). В любой из перечисленных позиций Петя может выиграть, утроив количество камней во второй куче.

Таким образом, ответ — 1619.

Задание 8

Укажите наибольшее десятичное число, при вводе которого на экране сначала напечатается 3, а затем 6.

Паскаль	**Алгоритмический язык**

var x, L, M: integer;
begin
  readln(x);
  L:=0; M:=0;
  while x > 0 do begin
  L:=L + 1;
  if (x mod 2) <> 0 then
  M:= M + x mod 8;
  x:= x div 8;
  end;
  writeln(L); write(M);
end.
	алг
	нач
	  цел x, L, M
	  ввод x
	  L := 0
	  M := 0
	  нц пока x > 0
	  L := L + 1
	  если mod(x,2) <> 0
	  то
	  M:= M + mod (x,8);
	  x := div(x,8)
	  все
	  кц
	  вывод L, нс, M
	кон

Си++	
#include <iostream> using namespace std; int main(void) {	

```
  int L, M, x;
  cin >> x;
   L = 0; M = 0;
  while (x > 0) {
      L = L + 1;
    if (x % 2 != 0) {
        M = M + x % 8;
    }
    x = x / 8;
  }
   cout << L << " " << M;
}
```

Решение

Необходимо получить трёхзначное восьмеричное число, у которого сумма нечётных разрядов равна 6. Следовательно, две последние цифры выглядят так: 51. А четный разряд (2) должен быть чётным максимальным в восьмеричной системе. Таким образом число, которое мы ищем $651_8 = 425_{10}$.

Ответ: 425.

Задание 9

Исполнитель А16 преобразует число, записанное на экране.

У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 2
3. Умножить на 2

Первая из них увеличивает число на экране на 1, вторая увеличивает его на 2, третья умножает его на 2.

Программа для исполнителя А16 – это последовательность команд.

Сколько существует таких программ, которые исходное число 3 преобразуют в число 12 и при этом траектория вычислений программы содержит число 10?

Траектория вычислений программы — это последовательность результатов выполнения всех команд программы. Например, для программы 132 при исходном числе 7 траектория будет состоять из чисел 8, 16, 18.

Решение

Искомое количество программ равно произведению количества программ, получающих из числа 3 число 10, на количество программ, получающих из числа 10 число 12.

Пусть R(n) — количество программ, которые число 3 преобразуют в число n, а P(n) — количество программ, которые число 10 преобразуют в число n.

Для всех n > 5 верны следующие соотношения:

1. Если n не делится на 2, то тогда R(n) = R(n - 1) + R(n - 2), так как существует два способа получения n — прибавлением единицы или прибавлением двойки. Аналогично P(n) = P(n - 1) + P(n - 2)

2. Если n делится на 2, тогда R(n) = R(n - 1) + R(n - 2) + R(n / 2). Аналогично P(n) = P(n - 1) + P(n - 2) + P(n / 2)

Последовательно вычислим значения R(n):

R(3) = 1
R(4) = R(3) = 1
R(5) = R(4) + R(3) = 1 + 1 = 2
R(6) = R(5) + R(4) + R(3) = 2 + 1 + 1 = 4
R(7) = R(6) + R(5) = 4 + 2 = 6
R(8) = R(7) + R(6) + R(4) = 6 + 4 + 1 = 11
R(9) = R(8) + R(7) = 11 + 6 = 17
R(10) = R(9) + R(8) + R(5) = 17 + 11 + 2 = 30

Теперь вычислим значения P(n):

P(10) = 1
P(11) = P(10) = 1
P(12) = P(11) + P(10) = 2

Таким образом, количество программ, удовлетворяющих условию задачи, равно 30 · 2 = 60.

Ответ: 60.

Задание 10

Текстовый файл содержит строки различной длины. Общий объём файла не превышает 1 Мбайт. Строки содержат только заглавные буквы латинского алфавита (ABC…Z).

Необходимо найти строку, содержащую наименьшее количество букв N (если таких строк несколько, надо взять ту, которая находится в файле раньше), и определить, какая буква встречается в этой строке чаще всего. Если таких букв несколько, надо взять ту, которая позже стоит в алфавите.

Решение

Для решения этой задачи объявим массив arr, в котором будем считать количество каждого символа в строке, переменную min, в которой будем хранить минимальное найденное количество символов N в какой-либо строке, переменную countN, в которую будем записывать количество символов N в очередной считанной строке. Последовательно считывая строки, будем считать в них количество символов N и записывать это количество в пере-

менную countN. Также будем считать в очередной считанной строке количество каждого отдельного символа с помощью массива arr. Если значение переменной countN окажется меньше текущего значения переменной min, будем находить максимальное значение элемента массива arr и символ, соответствующий этому значению. Если в строке будет встречено одинаковое количество двух разных символов, необходимо в переменную res записать тот символ, который согласно алфавиту идёт позже. Для этого при поиске максимального значения элемента массива используем условие «maxC <= arr[j]».

Решение задачи на языке PascalABC

```
var
arr: array ['A'..'Z'] of integer;
i, min, countN, maxC: integer;
j, res: char;
str: string;
f: text;
begin
min := 100000;
maxC := 0;
countN := 0;
assign(f,'C:\24.txt');
reset(f);
for j := 'A' to 'Z' do arr[j] := 0;
while not eof(f) do begin
readln(f, str);
for i:= 1 to str.Length do begin
if (str[i] = 'N') then countN := countN + 1;
arr[str[i]] := arr[str[i]] + 1;
end;
if min > countN then begin
min := countN;
maxC := 0;
for j := 'A' to 'Z' do begin
if (maxC <= arr[j]) and (j <> 'N') then begin
maxC := arr[j];
res := j;
end;
end;
end;
for j := 'A' to 'Z' do arr[j] := 0;
countN := 0;
end;
writeln(res);
end.
```

В результате работы данного алгоритма при вводе данных из файла в условии получаем ответ — Y.

Ответ: Y.

Задание 11

Найдите все натуральные числа, принадлежащие отрезку [45 000 000; 50 000 000], у которых ровно пять различных нечётных делителей (количество чётных делителей может быть любым). В ответе перечислите найденные числа в порядке возрастания.

Решение

Решим задачу перебором. Находя очередной делитель числа, будем проверять, является ли этот делитель нечётным, и, если является, будем прибавлять к переменной count единицу. Также заметим, что у каждого делителя числа есть парный делитель — если он нечётный, будем прибавлять к переменой count единицу. Также учтём то, что у числа может быть 2 одинаковых делителя (например, у числа 9 два одинаковых делителя — числа 3 и 3).

Решение на языке Pascal

```
var
   count, i, j, k, sqrtI: longint;
begin
   for i := 45000000 to 50000000 do begin
      count := 0;
      sqrtI := round(sqrt(i));
      for j := 1 to sqrtI do begin
         if i mod j = 0 then begin
            if j mod 2 = 1 then count := count + 1;
            k := i div j;
            if k mod 2 = 1 then begin
               count:=count + 1;
               if j = k then count:=count - 1;
            end;
        end;
        if count > 5 then break;
      end;
      if count = 5 then writeln(i);
      end;
end.
```

В результате работы программа должна вывести следующее:

45212176
45265984
47458321
48469444

Задание 12

На грузовом судне необходимо перевезти контейнеры, имеющие одинаковый габарит и разные массы (некоторые контейнеры могут иметь одинаковую массу). Общая масса всех контейнеров превышает грузоподъёмность судна. Количество грузовых мест на судне не меньше количества контейнеров, назначенных к перевозке. Какое максимальное количество контейнеров можно перевезти за один рейс и какова масса самого тяжёлого контейнера среди всех контейнеров, которые можно перевезти за один рейс?

Входные данные

В первой строке входного файла находятся два числа: S — грузоподъёмность судна (натуральное число, не превышающее 100 000) и N — количество контейнеров (натуральное число, не превышающее 20 000). В следующих N строках находятся значения масс контейнеров, требующих транспортировки (все числа натуральные, не превышающие 100), каждое в отдельной строке.

Выходные данные

Два целых неотрицательных числа: максимальное количество контейнеров, которые можно перевезти за один рейс и масса наиболее тяжёлого из них.

Пример входного файла:

100 4
80
30
50
40

При таких исходных данных можно транспортировать за один раз максимум два контейнера. Возможные массы этих двух контейнеров 30 и 40, 30 и 50 или 40 и 50. Поэтому ответ для приведённого примера: 2 50.

Решение

Сначала считаем в массив данные из файла. После этого отсортируем массив в порядке возрастания. Таким образом, последовательно складывая элементы массива с начала и сравнивая сумму с оставшейся грузоподъёмностью получим максимальное количество контейнеров, которые можно перевезти. Далее, вычитая из найденной суммы наибольшую массу контейнера в текущей последовательности, будем пробовать прибавлять контейнеры с большей массой. Если такой контейнер будет найден, то заменяем значение контейнера с наибольшей массой, который возможно поместить на судно.

Решение на языке Pascal

```
var
i, j, t: integer;
a: array [1..10000] of integer;
s: integer;
```

```
n: integer;
sum: integer;
maxi: integer;
f: text;
begin
   assign(f,'C:\26.txt');
   reset(f);
   readln(f, s, n);
   for i := 1 to n do readln(f, a[i]);
   for i := 1 to n do
      for j := i + 1 to n do
         if a[i] > a[j] then begin
            t := a[i];
            a[i] := a[j];
            a[j] := t;
         end;
   sum := 0;
   maxi := 1;
   for i := 1 to n do
      if sum + a[i] <= s then begin
         sum := sum + a[i];
         maxi := i;
      end;
   t := a[maxi];
   for i := maxi to n do
      if ((sum - t) + a[i]) <= s then begin
         sum := sum - t + a[i];
         t := a[i];
      end;
   writeln(maxi, ' ', t);
end.
```

В результате работы данного алгоритма при вводе данных из файла в условии получаем ответ — 1612 90.

Ответ: 1612 90.

Задание 13

Имеется набор данных, состоящий из троек положительных целых чисел. Необходимо выбрать из каждой тройки ровно одно число так, чтобы сумма всех выбранных чисел не делилась на $k = 109$ и при этом была максимально возможной. Гарантируется, что искомую сумму получить можно. Программа должна напечатать одно число — максимально возможную сумму, соответствующую условиям задачи.

Входные данные.

Даны два входных файла (файл *A* и файл *B*), каждый из которых содержит в первой строке количество троек N ($1 \le N \le 1\,000\,000$). Каждая из следующих N строк содержит три натуральных числа, не превышающих 20 000.

Пример организации исходных данных во входном файле:

```
6
1 3 7
5 12 6
6 9 11
5 4 8
3 5 4
1 1 1
```

Для указанных входных данных, в случае, если $k = 5$, значением искомой суммы является число 44.

В ответе укажите два числа: сначала значение искомой суммы для файла *A*, затем для файла *B*.

Решение

Будем последовательно считывать тройки чисел из файла и прибавлять к переменной sum максимальное число в тройке. Также будем находить минимальную разницу между максимальным числом в тройке и минимальным числом в тройке, между максимальным числом в тройке и средним по значению числом в тройке. Разница между числами при этом не должна делиться на 109 без остатка. В конце выполнения программы будем проверять, делится ли найденная сумма на 109 без остатка, и если не делится — будем выводить найденную сумму, иначе будем выводить найденную сумму, вычтя из неё найденную минимальную разницу между числами.

Решение задачи на языке Pascal

```
var
x, y, z: integer;
n: integer;
sum: int64;
minDif: integer;
f: text;
begin
assign(f,'C:\27_B.txt');
reset(f);
readln(f, n);
sum := 0;
minDif := 40001;
while not eof(f) do begin
readln(f, x, y, z);
if (x >= y) and (x >= z) then begin
sum := sum + x;
```

```
if (((abs(x - y)) mod 109 <> 0) and (abs(x - y) < minDif)) then
minDif := abs(x - y);
if (((abs(x - z)) mod 109 <> 0) and (abs(x - z) < minDif)) then
minDif := abs(x - z);
end
else if (y >= z) and (y >= x) then begin
sum := sum + y;
if (((abs(y - x)) mod 109 <> 0) and (abs(y - x) < minDif)) then
minDif := abs(y - x);
if (((abs(y - z)) mod 109 <> 0) and (abs(y - z) < minDif)) then
minDif := abs(y - z);
end
else if (z >= x) and (z >= y) then begin
sum := sum + z;
if (((abs(z - y)) mod 109 <> 0) and (abs(z - y) < minDif)) then
minDif := abs(z - y);
if (((abs(z - x)) mod 109 <> 0) and (abs(z - x) < minDif)) then
minDif := abs(z - x);
end;
end;
if sum mod 109 <> 0 then
writeln(sum)
else writeln(sum - minDif);
end.
```

3. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная литература

1. Кудрина, Е.В. Основы алгоритмизации и программирования на языке C#: учеб. пособие для СПО / Е.В. Кудрина, М.В. Огнева. – М.: Издательство Юрайт, 2019. – 322 с.

2. Трофимов, В.В. Основы алгоритмизации и программирования: учебник для СПО / В.В. Трофимов, Т.А. Павловская; под ред. В.В. Трофимова. – М.: Издательство Юрайт, 2019. – 137 с.

3. Федоров, Д.Ю. Программирование на языке высокого уровня Python: учеб. пособие для СПО / Д.Ю. Федоров. – М.: Издательство Юрайт, 2019. – 126 с.

Дополнительная литература

1. Кубенский, А.А. Функциональное программирование: учебник и практикум для академического бакалавриата / А. А. Кубенский. – М.: Издательство Юрайт, 2019. – 348 с.

2. Нагаева, И.А. Программирование: delphi: учеб. пособие для академического бакалавриата / И.А. Нагаева, И.А. Кузнецов; под ред. И.А. Нагаевой. – М.: Издательство Юрайт, 2017. – 302 с.