Всероссийская олимпиада школьников
«Миссия выполнима. Твое призвание-финансист!»

ЗАДАНИЯ, РЕШЕНИЯ, КРИТЕРИИ
ЗАКЛЮЧИТЕЛЬНЫЙ (ОЧНЫЙ) ЭТАП
Информатика 10-11 класс, 2016/2017 учебный год

Инструкция для участника олимпиады

Организационные указания

Продолжительность олимпиады – 180 минут (3 астрономических часа). Олимпиадное задание состоит из четырех задач. Решение первой задачи связано с определением ответа, решение задач 2-4 предполагает разработку программ. Для каждой задачи указан ее вес. Участник олимпиады самостоятельно определяет последовательность выполнения задач. Участник олимпиады получает бумагу для черновика.

Для разработки программ участник олимпиады должен выбрать один из предлагаемых языков программирования: C/C++, C#, Basic, Pascal, Python. Порядок запуска выбранной среды программирования разъясняет член комиссии перед началом работы.

Перед началом выполнения задания создайте на рабочем столе компьютера папку с именем следующего формата:

Олимпиада_*КодУчастника*

например, **Олимпиада_5034**

Внутри указанной папки Вы должны представить разработанные программы в формате выбранной среды программирования.

Первой строкой каждой программы должен быть комментарий в формате:

КодУчастника, **задача** *N*

например, для С-подобных языков:

// 5034, задача 1

Доступ к сети Интернету отключается. Исключается использование участником любых приемо-передающих устройств.

Сообщите члену комиссии об окончании ответа на задание и ждите его указаний.

Вычисление итоговой оценки

Каждая программа, разработанная участником олимпиады, оценивается членом жюри, исходя из 100 баллов. Если программа содержит недоработки или ошибки, то оценка снижается.

Каждую программу проверяет несколько членов жюри. Итоговая оценка $И_i$, выставляемая i-тым членом жюри, получается как сумма оценок, умноженных на соответствующий весовой коэффициент (Б*К). Итоговая оценка ИТОГО, выставляемая участнику всеми членами жюри, вычисляется как средняя арифметическая величина. Таким образом:

$$И_i = Б_1 * К_1 + Б_2 * К_2 + Б_3 * К_3 + Б_4 * К_4,$$

где:

Б – оценка за программу по 100-бальной системе;

К – весовой коэффициент. $К_1 + К_2 + К_3 + К_4 = 1.0$

$К_1 = 0{,}10$

$К_2 = 0{,}20$

$К_3 = 0{,}25$

$К_4 = 0{,}45$

Итоговая оценка выставляется по следующей формуле:

$$ИТОГО = (И_1 + И_{2+} \ldots И_n)/n$$

где: n – количество членов жюри, проверяющих программы.

$И_i$ – оценка, выставленная i-тым членом жюри.

Критерии оценивания программы

Поскольку разработка программы является творческим процессом, то и оценивание программы, в значительной степени, носит субъективный характер. Тем не менее, оценки, выставляемые разными членами жюри, как правило, имеют небольшой разброс, а оценивание несколькими членами жюри позволяет получить итоговую оценку, близкую к истинной.

Каждая программа оценивается по 100-бальной шкале в соответствии с показателями, перечисленными в следующей таблице:

Показатели оценивания	**Балл ($ПР_{100}$)**
1. Программа полностью соответствует условию, выполняет поставленную задачу и ни при каких обстоятельствах не завершается аварийно. 2. Пользовательский интерфейс эргономичен и интуитивно понятен. 3. Алгоритмы обработки данных эффективны и рациональны. 4. Обнаружены несущественные недостатки.	86-100
Не выполнены требования на оценку «отлично», при этом выявлены один или несколько недостатков: 1. Программа в основном соответствует условию задачи. Допущены несущественные отклонения от условия. 2. Программа завершается аварийно только при вводе некорректных данных или выполнении второстепенных функций. 3. Пользовательский интерфейс недостаточно эргономичен и интуитивно непонятен. 4. Алгоритмы обработки данных неэффективны. Существует более простой способ решения задачи.	70-85
Не выполнены требования на оценку «хорошо», при этом: 1. Программа имеет отклонения от условия. Задача решена частично. 2. Отдельные фрагменты не отлажены до конца, но в целом близки к правильному алгоритму.	50-69
1. Работу не удается идентифицировать или работа не найдена. 2. Программа имеет синтаксические ошибки. 3. Программа в целом не соответствует условию.	0-49

Участник олимпиады должен обратить внимание на следующие показатели качества программ:

- Отсутствие синтаксических ошибок;
- Отсутствие аварийных завершений программы;
- Соответствие программы условию задачи;
- Полнота отлаженности программы, вывод правильного ответа для любых допустимых исходных данных;
- Качество интерфейса;
- Эффективность и рациональность алгоритмов;
- Контроль вводимых с клавиатуры исходных данных;
- Читабельность программы;
- Наличие комментариев;
- Возможность повторного выполнения программы для новых исходных данных.

За неработающую программу член жюри может выставить некоторые баллы, если в программе частично реализован алгоритм, близкий к правильному.

Если отсутствует контроль вводимых с клавиатуры исходных данных, то 100 бальная оценка Б уменьшается до 8 баллов. За отсутствие комментариев оценка уменьшается на 2 балла. За плохую читабельность оценка уменьшается на 2 балла. За отсутствие повторения программы для новых исходных данных оценка уменьшается на 2 балла.

Если программа имеет синтаксические ошибки, оценка уменьшается не менее чем на 50 баллов.

Если программа содержит нерациональные и неэффективные, но работающие алгоритмы, оценка может быть снижена до 20 баллов.

Если программа для отдельных комбинаций исходных данных завершается аварийно, а для других – нет, оценка может быть снижена до 40 баллов.

Неполное соответствие программы условию задачи, неполная отлаженность программы, невысокое качество интерфейса и т.д. приводят к снижению оценки на величину, определяемую степенью недоработки программы.

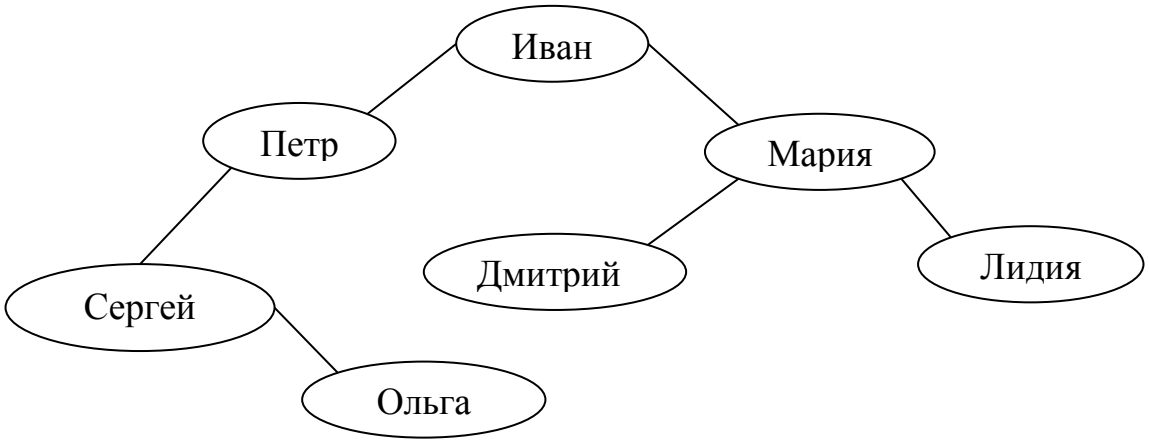
Критерии определения победителей и призеров заключительного этапа олимпиады

Участник олимпиады, набравший наибольшее количество баллов, является победителем олимпиады. Следующие по количеству баллов участники получают соответственно второе и третье место. Участник участвует в конкурсе, если он набрал больше 49 баллов.

ВАРИАНТ 1

Задача 1. Вес 10 баллов

На рисунке приведена родословная человека по имени Иван в виде генеалогического дерева. Каждый узел дерева содержит имя человека и ссылку на левое и правое поддерево. Данные об именах некоторых предков могут отсутствовать.



Имена, указанные в родословной, выводятся по правилу, определенному процедурой:

```
Процедура Просмотр (дерево)
    Начало
      Если дерево пустое То выход
      Если левое поддерево есть То
          Просмотр (левое поддерево)
          Вывод имени в узле левого поддерева
      Если правое поддерево есть То
         Просмотр (правое поддерево)
    Конец
```

Определите последовательность имен, которые будут выведены по этому алгоритму для человека по имени ИВАН. Создайте файл Задача1.txt и запишите в него ответ.

Правильный ответ:

Сергей Петр Дмитрий

Правильный ответ может быть подтвержден с помощью следующей программы на языке C#.

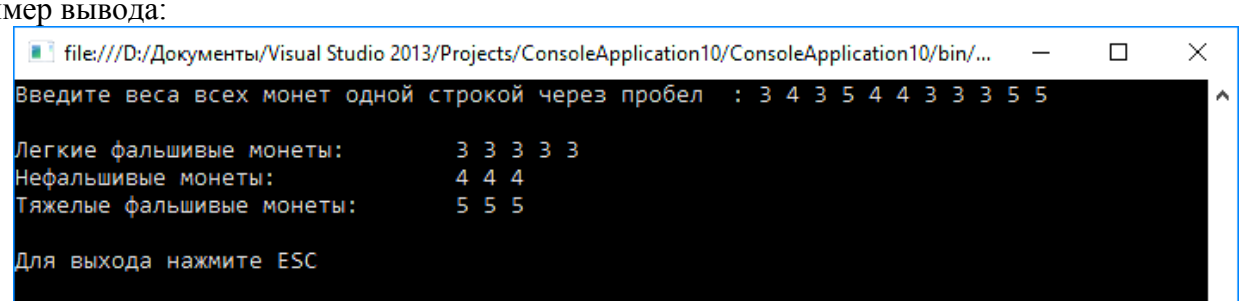
```
class Узел
{
   public string текст;
   public Узел левое;
   public Узел правое;

   public Узел(string т, Узел л, Узел п)
   {
      текст = т;  левое = л;  правое = п;
   }
}

class Program
{
   static void Просмотр(Узел дерево)
   {
      if (дерево.левое == null && дерево.правое == null) return;
      if (дерево.левое != null)
      {
         Просмотр(дерево.левое);
```

```
            System.Console.WriteLine (дерево.левое.текст);
        }
        if (дерево.правое != null)
            Просмотр(дерево.правое);
    }

    static void Main()
    {
        Узел Ольга   = new Узел("Ольга", null, null);

        Узел Сергей  = new Узел("Сергей", null, Ольга);
        Узел Петр    = new Узел("Петр", Сергей, null);

        Узел Дмитрий = new Узел("Дмитрий", null, null);
        Узел Лидия   = new Узел("Лидия", null, null);
        Узел Мария   = new Узел("Мария", Дмитрий, Лидия);

        Узел Иван    = new Узел("Иван", Петр, Мария);

        Просмотр(Иван);
    }
}
```

Задача 2. Вес 20 баллов

С клавиатуры вводятся веса N монет (N>0), среди которых может быть Z законных монет (Z>0), W более легких фальшивых монет и R более тяжелых фальшивых монет, причем фальшивые монеты как одного, так и другого веса могут отсутствовать (0<=W<N, 0<=R<N). Все легкие фальшивые монеты имеют одинаковый вес, все тяжелые фальшивые монеты так же имеют одинаковый вес. Если присутствуют монеты только двух весов, то считается, что отсутствуют тяжелые фальшивые монеты. Требуется вывести на экран сначала более легкие фальшивые монеты, затем законные и далее – более тяжелые фальшивые монеты. При оценивании программы предпочтение будет отдано простоте и оригинальности алгоритма.

Пример вывода:



Примерное решение задачи 2 на языке C#:

Необходимо обратить внимание на простоту и оригинальность решения задачи. Например, решение получается более компактным, если веса монет представить в виде массива и затем массив отсортировать.

```
using System;

class Program
```

```
{
   static void Main(string[] args)
   {
      do
      {
         bool fl = false;
         char[] sep = { ' ' };                    // Разделитель
         string s;
         int[] m = null;
         int N = 0;
         do
         {
            // Ввод массива монет с клавиатуры.
            Console.Write("Введите веса всех монет одной строкой через пробел  : ");
            s = Console.ReadLine();
            string[] mstr = s.Split(sep, StringSplitOptions.RemoveEmptyEntries);
            N = mstr.Length;
            if (N == 0) { Console.Write("Вы не ввели веса монет. Повторите ввод."); continue; }

            // Преобразование с проверкой
            m = new int[mstr.Length];
            for (int f = 0; f < N; f++)
            {
               if (int.TryParse(mstr[f], out m[f]) == false)
               {
                  Console.WriteLine("Неверный формат введенного числа: {0}. Повторите ввод.",
                                    mstr[f]);  break;
               }
               fl = true;
            }

         } while (!fl);

         Sort(m);

         int n1 = 0, n2 = 0, n3 = 0;

         int i, j, k;
         n1 = 1;
         for (i = 0; i < N - 1; i++)
            if (m[i] == m[i + 1]) n1++; else break;

         if (i != N - 1)
         {
            n2 = 1;
            for (j = i + 1; j < N - 1; j++)
               if (m[j] == m[j + 1]) n2++; else break;

            if (j != N - 1)
            {
               n3 = 1;
               for (k = j + 1; k < N - 1; k++)
                  if (m[k] == m[k + 1]) n3++; else break;
```

```
            if (k != N - 1) { Console.WriteLine(
                  "Вы ввели более трех различных весов монет. Повторите ввод."); continue; }
            }
         }

         //Console.WriteLine("Количества монет: {0}, {1}, {2}", n1, n2, n3);
         string легкие = "", тяжелые = "";
         if (n2 == 0)
         {
            n2 = n1; n1 = 0; легкие = "НЕТ";
         }
         if (n3 == 0) тяжелые = "НЕТ";

         Console.Write("\nЛегкие фальшивые монеты:  \t" + легкие);
         for (i = 0; i < n1; i++)
            Console.Write(m[i].ToString() + " ");

         Console.Write("\nНефальшивые монеты:       \t");
         for (i = n1; i < n1+n2; i++)
            Console.Write(m[i].ToString() + " ");

         Console.Write("\nТяжелые фальшивые монеты: \t" + тяжелые);
         for (i = n1+n2; i < N; i++)
            Console.Write(m[i].ToString() + " ");


         Console.WriteLine("\n\nДля выхода нажмите ESC ");
      } while (Console.ReadKey(true).Key != ConsoleKey.Escape);

   }

   //---------------------------------------
   static void Sort(int[] m)
   {
      int temp; // Для запоминания элемента

      for (int i = 0; i < m.Length - 1; i++)
         // На место элемента i всплывает самый меньший
         for (int j = i + 1; j < m.Length; j++)
            if (m[i] > m[j])                // Поменять местами
            {
               temp = m[j];
               m[j] = m[i];
               m[i] = temp;
            }
   }

}
```

Задача 3. Вес 25 баллов

Создать матрицу A целых чисел размером MxN и заполнить ее случайными положительными двузначными числами. Значения M и N вводятся с клавиатуры. Определить локальные минимумы в матрице. Локальным минимумом является элемент матрицы, меньший соседних элементов по горизонтали, вертикали и диагоналям. Граничные элементы матрицы рассматривать на соответствие локальному минимуму. Исходную матрицу, локальные минимумы с их номерами строк и столбцов вывести на экран.

При оценивании работы предпочтение будет отдано простоте и оригинальности алгоритма.

Пример вывода минимумов:

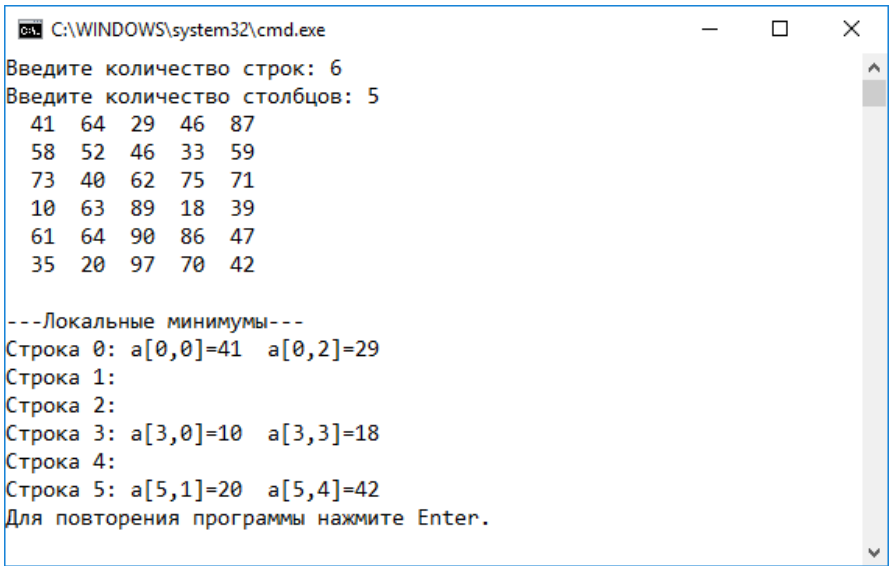
```
. . .
---Локальные минимумы---
Строка 0:
Строка 1: A[1,2]=13   A[1,4]=28
. . .
```

Примерное решение задачи 3 на языке C#:

```
using System;

class Program
{
   static void Main()
   {
      int m, n;            // Количество строк и столбцов
      int[,] a;            // Ссылка на формируемую матрицу

      int i, j;            // Номер строки и  номер столбца

      do
      {
         // Ввод с проверкой
         Console.Write("Введите количество строк: ");
         while (!int.TryParse(Console.ReadLine(), out m) || m < 1)
            Console.WriteLine("Вы ошиблись при вводе. Повторите!");

         Console.Write("Введите количество столбцов: ");
         while (!int.TryParse(Console.ReadLine(), out n) || n < 1)
            Console.WriteLine("Вы ошиблись при вводе. Повторите!");

         a = new int[m+2, n+2];  // расширенная матрица


         // Инициализация расширенной матрицы

         for (i = 0; i < m + 2; i++)
            for (j = 0; j < n + 2; j++)
               a[i, j] = 100;

         // Инициализация матрицы
         Random ran = new Random();
         for (i = 1; i < m + 1; i++, Console.WriteLine())
            for (j = 1; j < n + 1; j++)
            {
```

```
                a[i, j] = ran.Next(10, 100);
                Console.Write("{0,4}", a[i, j]);
            }


        // Поиск локальных минимумов

        Console.WriteLine("\n---Локальные минимумы---");

        for (i = 1; i < m + 1; i++, Console.WriteLine())
        {
            Console.Write("Строка {0}: ", (i - 1));
            for (j = 1; j < n + 1; j++)
                if (   a[i, j] < a[i-1, j-1]
                    && a[i, j] < a[i, j-1]
                    && a[i, j] < a[i+1, j-1]

                    && a[i, j] < a[i-1, j]
                    && a[i, j] < a[i+1, j]

                    && a[i, j] < a[i-1, j+1]
                    && a[i, j] < a[i, j+1]
                    && a[i, j] < a[i+1, j+1])
                {
                    Console.Write("a[{0},{1}]={2}  ", i-1, j-1, a[i, j]);
                }
        }

        Console.WriteLine("Для повторения программы нажмите Enter.\n\n");
        Console.WriteLine("----------------------------------------");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);
  }
}
```

Вывод:



Необходимо обратить внимание на простоту и оригинальность решения задачи. Например, решение значительно упрощается, если исходную матрицу по её контуру дополнить строками и

столбцами (по 1 с каждой стороны) и заполнить новые элементы заведомо большими значениями, не влияющими на локальные минимумы. В такой расширенной матрице анализ граничных элементов исходной матрицы не требует специальных циклов и условий и осуществляется одним вложенным циклом.

Программа должна работать для любых значения N и M.

Задача 4. Вес 45 баллов

На игровом поле слева-направо расположено N черных фишек, затем пустая ячейка и далее N белых фишек.

Требуется передвинуть черные фишки на место белых, а белые - на место черных за наименьшее число ходов. Фишку можно передвигать либо в соседнюю с ним пустую ячейку, либо в пустую ячейку, которая находится непосредственно за ближайшей фишкой. Причем, белые фишки разрешается двигать только влево, а черные - только вправо. Последовательность ходов произвольная.

Разработать программу для отображения процесса перемещений фишек до конечного состояния. После каждого перемещения какой-либо фишки игровое поле должно быть выведено на экран. В конце решения задачи вывести информацию о количестве выполненных перемещений.

Каждую черную фишку обозначить символом '*', каждую белую - символом 'o'. Для отображения пустой ячейки использовать символ '_'.

Число N ввести с клавиатуры.

Пример вывода начального состояния игрового поля и первого хода (для M=4):

****_oooo
***_*oooo

Примерное решение задачи 4 на языке C#:

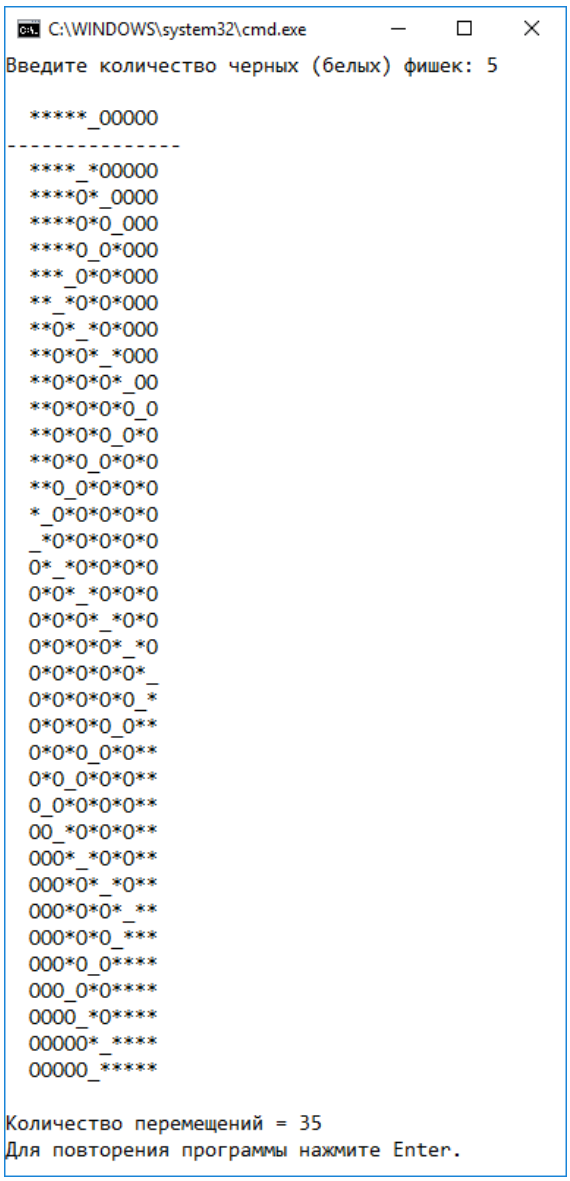
```
using System;

class Program
{
   static int count = 0;

   static void Main(string[] args)
   {
      do
      {
         int n;      // Количество черных (белых) фишек

         Console.Write("Введите количество черных (белых) фишек: ");
         while (!int.TryParse(Console.ReadLine(), out n) || n < 1)
            Console.WriteLine("Вы ошиблись при вводе. Повторите!");

         char[] a = new char[n + n + 1];     // Игровое поле
         n = a.Length - 1;                   // Индекс правой границы поля
         int i = n / 2;                      // Позиция пустой ячейки

         // Инициализация игрового поля
         for (int j = 0; j < n; j++)
            a[j] = '*';         // Черные фишки
```

```
a[i] = '_';            // Пустая ячейка

for (int j = i + 1; j < a.Length; j++)
   a[j] = 'O';       // Белые фишки

int c = 1;             // Цвет фишки: 1-черный, -1 -белый
int k = 2;             // Индекс левой границы поля

Console.WriteLine();
Print(a);
Console.WriteLine("--" + new string('-', n+3 ));

while (i != 0 && i != n)
{
   if (a[i - 1] != a[i + 1])
   {
      // Пара разного цвета
      a[i] = a[i - c];        // Двигаем
      i = i - c;
      a[i] = '_';
      c *= -1;                // Меняем цвет
      Print(a);
   }
   else
   {
      // Пара одного цвета
      a[i] = a[i - c * 2];    // Перескакиваем
      i = i - c * 2;
      a[i] = '_';
      Print(a);
   }
}

//.....................................
while (k <= n + 1)
{
   if (i == k - 2 && c == -1 && a[i + 1] == 'O')
   {
      // Левый край поля
      a[k - 2] = a[k - 1];        // Сдвигаем
      i = k - 1;
      a[i] = '_';
      k++;
      Print(a);
   }

   if (i == n && c == 1 && a[n - 1] == '*')
   {
      // Правый край поля
      a[n] = a[n - 1];        // Двигаем
      i = n - 1;
      a[i] = '_';
      n--;
```

```
                Print(a);
            }
            //.....................................
            if (c == 1)
            {               // Перемещаем чер.фишки, начиная с правого края
                while (i >= k)
                {
                    a[i] = a[i - 2];    // Перескакиваем
                    i = i - 2;
                    a[i] = '_';         // Пустая ячейка двигается влево
                    Print(a);
                }

                c = -1;               // Меняем цвет на белый
                n--;
                continue;
            }

            //.....................................
            if (c == -1)
            {               // Перемещаем белые фишки, начиная с левого края
                while (i <= n - 2)
                {
                    a[i] = a[i + 2];    // Перескакиваем
                    i = i + 2;
                    a[i] = '_';
                    Print(a);
                }
                c = 1;              // Меняем цвет на черный
                k++;
                continue;
            }
        }
        Console.WriteLine("\nКоличество перемещений = " + (count-1));
        count = 0;

        Console.WriteLine("Для повторения программы нажмите Enter.\n\n");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);
}

static void Print(char[] a)
{
    count++;
   Console.Write("  ");
    foreach (var e in a)
        Console.Write(e.ToString());
    Console.WriteLine();
}
}
```

Для успешного решения задачи можно придерживаться следующего правила: на первом этапе перемещаемая фишка не должна примыкать к фишке того же цвета, если только все фишки этого цвета не примыкают к границе.

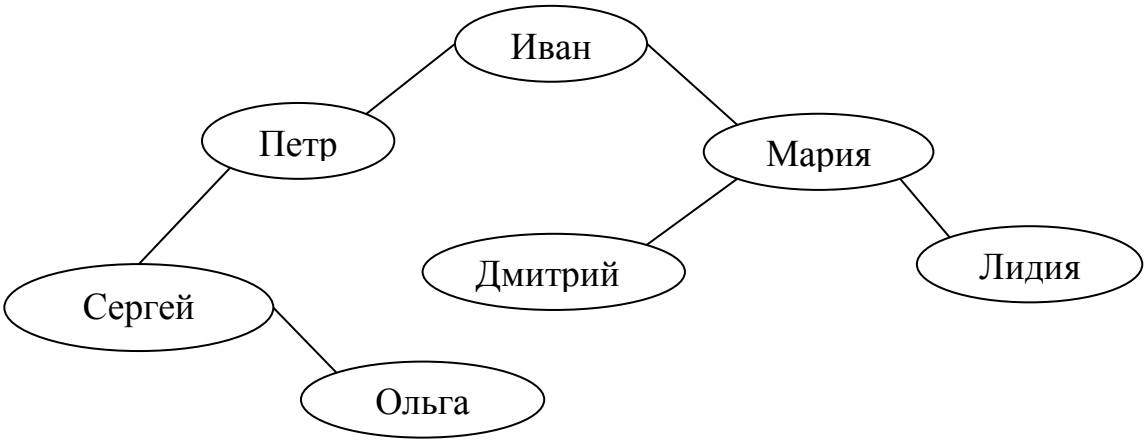
Пример вывода для N=5.



ВАРИАНТ 2

Задача 1. Вес 10 баллов

На рисунке приведена родословная человека по имени Иван в виде генеалогического дерева. Каждый узел дерева содержит имя человека и ссылку на левое и правое поддерево. Данные об именах некоторых предков могут отсутствовать.



Имена, указанные в родословной, выводятся по правилу, определенному процедурой:

```
Процедура Просмотр (дерево)
    Начало
       Если дерево пустое То выход
       Если левое поддерево есть То
            Просмотр (левое поддерево)
       Если правое поддерево есть То
           Просмотр (правое поддерево)
           Вывод имени в узле правого поддерева
    Конец
```

Определите последовательность имен, которые будут выведены по этому правилу для человека по имени ИВАН.

Правильный ответ:

Ольга Лидия Мария

Правильный ответ может быть подтвержден с помощью следующей программы на языке C#.

```
class Узел
{
   public string текст;
   public Узел левое;
   public Узел правое;

   public Узел(string т, Узел л, Узел п)
   {
      текст = т;  левое = л;  правое = п;
   }
}

class Program
{
   static void Просмотр(Узел дерево)
   {
      if (дерево.левое == null && дерево.правое == null) return;
      if (дерево.левое != null)
         Просмотр(дерево.левое);
```

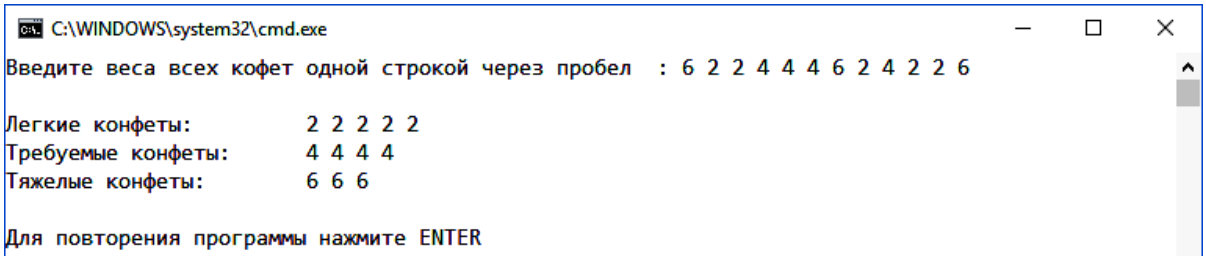
```
        if (дерево.правое != null)
        {
            Просмотр(дерево.правое);
            System.Console.WriteLine (дерево.правое.текст);
         }
    }

    static void Main()
    {
        Узел Ольга    = new Узел("Ольга", null, null);

        Узел Сергей   = new Узел("Сергей", null, Ольга);
        Узел Петр     = new Узел("Петр", Сергей, null);

        Узел Дмитрий = new Узел("Дмитрий", null, null);
        Узел Лидия    = new Узел("Лидия", null, null);
        Узел Мария    = new Узел("Мария", Дмитрий, Лидия);

        Узел Иван     = new Узел("Иван", Петр, Мария);

        Просмотр(Иван);
    }
}
```

Задача 2. Вес 20 баллов

При укладке N конфет в коробку случайно могли быть подмешаны два вида конфет с другой начинкой. Все конфеты не отличаются друг от друга по внешнему виду, но имеют разный вес. Конфеты с одной начинкой легче требуемых конфет, а конфеты с другой начинкой – тяжелее требуемых конфет. Требуется извлечь подмешанные конфеты.

Для разработки соответствующей программы ввести с клавиатуры веса N конфет (N>0), среди которых может быть Z требуемых конфет (Z>0), W более легких конфет и R более тяжелых конфет, причем подмешанные конфеты как одного, так и другого веса могут отсутствовать (0<=W<N, 0<=R<N). Если присутствуют конфеты только двух весов, то считается, что отсутствуют тяжелые конфеты. Требуется вывести на экран сначала более легкие конфеты, затем требуемые и далее – более тяжелые конфеты. При оценивании программы предпочтение будет отдано простоте и оригинальности алгоритма.

Пример вывода:



Примерное решение задачи 2 на языке C#:

Необходимо обратить внимание на простоту и оригинальность решения задачи. Например, решение получается более компактным, если веса конфет представить в виде массива, а затем массив отсортировать.

```
// Задача на конфеты
using System;

class Program
{
   static void Main(string[] args)
   {
      do
      {
         bool fl = false;
         char[] sep = { ' ' };                    // Разделитель
         string s;
         int[] m = null;
         int N = 0;
         do
         {
            // Ввод массива конфет с клавиатуры.
            Console.Write("Введите веса всех конфет одной строкой через пробел  : ");
            s = Console.ReadLine();
            string[] mstr = s.Split(sep, StringSplitOptions.RemoveEmptyEntries);
            N = mstr.Length;
            if (N == 0) { Console.Write("Вы не ввели веса конфет. Повторите ввод.");
                          continue; }

            // Преобразование с проверкой
            m = new int[mstr.Length];
            for (int f = 0; f < N; f++)
            {
               if (int.TryParse(mstr[f], out m[f]) == false)
               {
                  Console.WriteLine(
                        "Неверный формат введенного числа: {0}. Повторите ввод.",
                          mstr[f]); break;
               }
               fl = true;
            }

         } while (!fl);

         Sort(m);

         int n1 = 0, n2 = 0, n3 = 0;

         int i, j, k;
         n1 = 1;
         for (i = 0; i < N - 1; i++)
            if (m[i] == m[i + 1]) n1++; else break;

         if (i != N - 1)
```

```
        {
            n2 = 1;
            for (j = i + 1; j < N - 1; j++)
                if (m[j] == m[j + 1]) n2++; else break;

            if (j != N - 1)
            {
                n3 = 1;
                for (k = j + 1; k < N - 1; k++)
                    if (m[k] == m[k + 1]) n3++; else break;

                if (k != N - 1)
                {
                    Console.WriteLine(
                        "Вы ввели более трех различных весов конфет. Повторите ввод.");
                     continue;
                }
            }
        }

        //Console.WriteLine("Количества конфет: {0}, {1}, {2}", n1, n2, n3);
        string легкие = "", тяжелые = "";
        if (n2 == 0)
        {
            n2 = n1; n1 = 0; легкие = "НЕТ";
        }
        if (n3 == 0) тяжелые = "НЕТ";

        Console.Write("\nЛегкие конфеты:  \t" + легкие);
        for (i = 0; i < n1; i++)
            Console.Write(m[i].ToString() + " ");

        Console.Write("\nТребуемые конфеты: \t");
        for (i = n1; i < n1 + n2; i++)
            Console.Write(m[i].ToString() + " ");

        Console.Write("\nТяжелые конфеты: \t" + тяжелые);
        for (i = n1 + n2; i < N; i++)
            Console.Write(m[i].ToString() + " ");

        Console.WriteLine("\n\nДля повторения программы нажмите ENTER ");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);
}

static void Sort(int[] m)
{
    int temp; // Для запоминания элемента

    for (int i = 0; i < m.Length - 1; i++)
        // На место элемента i всплывает самый меньший
        for (int j = i + 1; j < m.Length; j++)
            if (m[i] > m[j])                // Поменять местами
            {
```

```
                temp = m[j];
                m[j] = m[i];
                m[i] = temp;
            }
    }
}
```

Задача 3. Вес 25 баллов

Создать двумерный массив M целых чисел размером n строк и k столбцов. Значения n и k вводятся с клавиатуры. Заполнить массив случайными положительными двузначными числами. Определить локальные максимумы в матрице. Локальным максимумом является элемент матрицы, больший соседних элементов по горизонтали, вертикали и диагоналям. Граничные элементы матрицы рассматривать на соответствие локальному максимуму. Исходную матрицу, локальные максимумы с их номерами строк и столбцов вывести на экран. При оценивании работы предпочтение будет отдано простоте и оригинальности алгоритма.

Пример вывода максимумов:

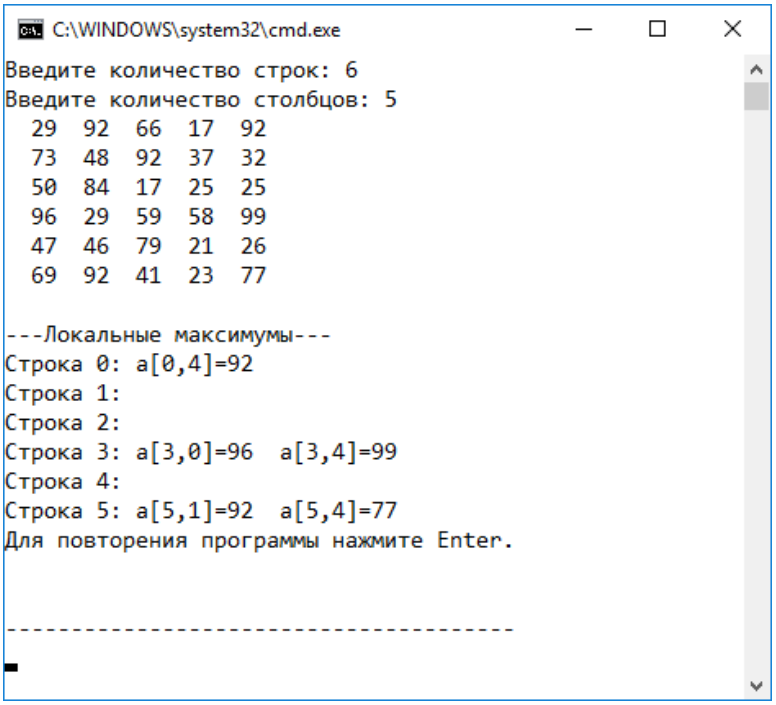
```
---Локальные максимумы---
Строка 0:
Строка 1: M[1,2]=98   M[1,4]=87
. . .
```

Примерное решение задачи 3 на языке C#:

```
using System;

class Program
{
   static void Main()
   {
      int m, n;             // Количество строк и столбцов
      int[,] a;             // Ссылка на формируемую матрицу

      int i, j;             // Номер строки и  номер столбца

      do
      {
         // Ввод с проверкой
         Console.Write("Введите количество строк: ");
         while (!int.TryParse(Console.ReadLine(), out m) || m < 1)
            Console.WriteLine("Вы ошиблись при вводе. Повторите!");

         Console.Write("Введите количество столбцов: ");
         while (!int.TryParse(Console.ReadLine(), out n) || n < 1)
            Console.WriteLine("Вы ошиблись при вводе. Повторите!");

         a = new int[m + 2, n + 2];  // расширенная матрица
```

```
        // Инициализация расширенной матрицы (для C# - необязательно)

        for (i = 0; i < m + 2; i++)
            for (j = 0; j < n + 2; j++)
                a[i, j] = 0;

        // Инициализация матрицы
        Random ran = new Random();
        for (i = 1; i < m + 1; i++, Console.WriteLine())
            for (j = 1; j < n + 1; j++)
            {
                a[i, j] = ran.Next(10, 100);
                Console.Write("{0,4}", a[i, j]);
            }


        // Поиск локальных максимумов

        Console.WriteLine("\n---Локальные максимумы---");

        for (i = 1; i < m + 1; i++, Console.WriteLine())
        {
            Console.Write("Строка {0}: ", (i - 1));
            for (j = 1; j < n + 1; j++)
                if (a[i, j] > a[i - 1, j - 1]
                    && a[i, j] > a[i, j - 1]
                    && a[i, j] > a[i + 1, j - 1]

                    && a[i, j] > a[i - 1, j]
                    && a[i, j] > a[i + 1, j]

                    && a[i, j] > a[i - 1, j + 1]
                    && a[i, j] > a[i, j + 1]
                    && a[i, j] > a[i + 1, j + 1])
                {
                    Console.Write("a[{0},{1}]={2}  ", i - 1, j - 1, a[i, j]);
                }
        }


        Console.WriteLine("Для повторения программы нажмите Enter.\n\n");
        Console.WriteLine("----------------------------------------");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);
  }
}
```

Вывод:



Необходимо обратить внимание на простоту и оригинальность решения задачи. Например, решение значительно упрощается, если исходную матрицу по её контуру дополнить строками и столбцами (по 1 с каждой стороны) и заполнить новые элементы заведомо большими значениями, не влияющими на локальные минимумы. В такой расширенной матрице анализ граничных элементов исходной матрицы не требует специальных циклов и условий и осуществляется одним вложенным циклом.

Программа должна работать для любых значения N и M.

Задача 4. Вес 45 баллов

В подряд расположенных лузах слева-направо находится M белых шаров, затем пустая луза и далее M черных шаров. (Число M ввести с клавиатуры.) Требуется поменять местами белые шары с черными за наименьшее число передвижений. Шар можно передвигать либо в соседнюю с ним пустую лузу, либо в пустую лузу, которая находится непосредственно за ближайшим шаром. Причем, белые шары разрешается двигать только вправо, а черные – только влево.

Разработать программу для отображения процесса перемещений шаров до конечного состояния. После каждого перемещения какого-либо шара игровое поле должно быть выведено на экран. В конце решения задачи вывести информацию о количестве выполненных перемещений.

Обозначить в программе каждый белый шар символом 'o', а каждый черный - символом '#'. Для отображения пустой лузы использовать символ ' '.

Пример вывода начального состояния игрового поля (для M=5):

ooooo #####

Примерное решение задачи 4 на языке C#:

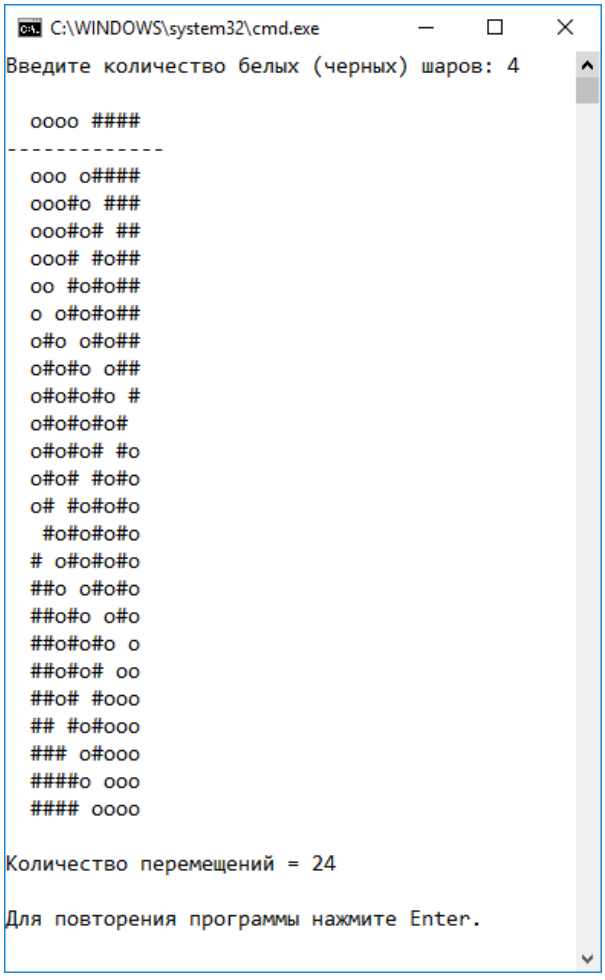
```
using System;

class Program
{
   static int count = 0;
```

```
static void Main(string[] args)
{
   do
   {
      int m;      // Количество черных (белых) шаров

      Console.Write("Введите количество белых (черных) шаров: ");
      while (!int.TryParse(Console.ReadLine(), out m) || m < 1)
         Console.WriteLine("Вы ошиблись при вводе. Повторите!");

      char[] a = new char[m + m + 1];     // Игровое поле
      m = a.Length - 1;                   // Индекс правой границы поля
      int i = m / 2;                      // Позиция пустой лузы

      // Инициализация игрового поля
      for (int j = 0; j < m; j++)
         a[j] = 'o';         // Белые шары

      a[i] = ' ';            // Пустая луза

      for (int j = i + 1; j < a.Length; j++)
         a[j] = '#';         // Черные шары

      int c = 1;             // Цвет шара: 1-белый, -1 -черный
      int k = 2;             // Индекс левой границы поля

      Console.WriteLine();
      Print(a);
      Console.WriteLine("--" + new string('-', m+3 ));

      while (i != 0 && i != m)
      {
         if (a[i - 1] != a[i + 1])
         {
            // Пара разного цвета
            a[i] = a[i - c];        // Двигаем
            i = i - c;
            a[i] = ' ';
            c *= -1;                // Меняем цвет
            Print(a);
         }
         else
         {
            // Пара одного цвета
            a[i] = a[i - c * 2];    // Перескакиваем
            i = i - c * 2;
            a[i] = ' ';
            Print(a);
         }
      }

      //.....................................
```

```
        while (k <= m + 1)
        {
            if (i == k - 2 && c == -1 && a[i + 1] == '#')
            {
                // Левый край поля
                a[k - 2] = a[k - 1];        // Сдвигаем
                i = k - 1;
                a[i] = ' ';
                k++;
                Print(a);
            }

            if (i == m && c == 1 && a[m - 1] == 'o')
            {
                // Правый край поля
                a[m] = a[m - 1];        // Двигаем
                i = m - 1;
                a[i] = ' ';
                m--;
                Print(a);
            }

            //.....................................
            if (c == 1)
            {               // Перемещаем белые шары, начиная с правого края
                while (i >= k)
                {
                    a[i] = a[i - 2];    // Перескакиваем
                    i = i - 2;
                    a[i] = ' ';         // Пустая луза двигается влево
                    Print(a);
                }

                c = -1;                 // Меняем цвет на черный
                m--;
                continue;
            }

            //.....................................
            if (c == -1)        // Перемещаем черные шары, начиная с левого края
            {
                while (i <= m - 2)
                {
                    a[i] = a[i + 2];    // Перескакиваем
                    i = i + 2;
                    a[i] = ' ';
                    Print(a);
                }
                c = 1;                  // Меняем цвет на белый
                k++;
                continue;
            }
        }
```

```
        Console.WriteLine("\nКоличество перемещений = " + (count-1));
        count = 0;

        Console.WriteLine("\nДля повторения программы нажмите Enter.\n\n");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);

  }
  static void Print(char[] a)
  {
    count++;
    Console.Write("  ");
    foreach (var e in a)
      Console.Write(e.ToString());
    Console.WriteLine();
  }
}
```

Вывод для N=4.



Всероссийская олимпиада школьников
«Миссия выполнима. Твое призвание-финансист!»

ОТБОРОЧНЫЙ (ЗАОЧНЫЙ) ЭТАП
Информатика 10-11 класс, 2016/2017 учебный год

Время выполнения заданий - 180 минут.

Задания с выборкой ответа

1. В классе 32 ученика. По результатам сдачи экзаменов 7 учеников получили оценку "5", 9 получили "4", 15 учеников – "3" и один – "2". Укажите вариант формулы, которая позволяет определить полное количество информации в сообщении, выясняющем, получил ли положительную оценку на экзамене выбранный наугад из списка ученик. Обозначение log2(i) – это двоичный логарифм числа i.

1) -(7/32)log2(7/32)-(9/32)log2(9/32)-(15/32)log2(15/32)-(1/32)log2(1/32)
2) log2(32)
3) -(31/32)log2(31)+6
4) -(31/32)log2(31)+5
5) -log2(31/32)

2. Истинность двух высказываний: "неверно, что если школьник B идет в кино, то C не идет" и "неверно, что из двух школьников A и C в кино идет только один" означает, что в кино пойдут школьники?

1) B и C
2) B
3) A, B и C
4) A и B
5) A и C

3. Дана логическая функция F(x,y,z) = not(x and y or not x and z or y and z), где and, or, not – знаки логических операций конъюнкции, дизъюнкции и отрицания соответственно. Для этой функции построена типовая таблица истинности, в которой номера строк отсчитываются от нуля. Переменные в таблице истинности указываются слева направо в очередности: x,y,z . В первой строке таблицы эти переменные равны 0, в последней – 1. Укажите все номера строк таблицы истинности, где функция принимает значение 1 (истина).

1) 0,2,4,5
2) 1,3,6,7
3) 0,3,4,5
4) 0,2,4
5) 0,3,4,6

4. Результатом минимизации логической функции F=(not a and not b and not c) or (a and not b and not c) or (a and not b and c), где and, or, not – знаки логических операций конъюнкции, дизъюнкции и отрицания соответственно, является функция:

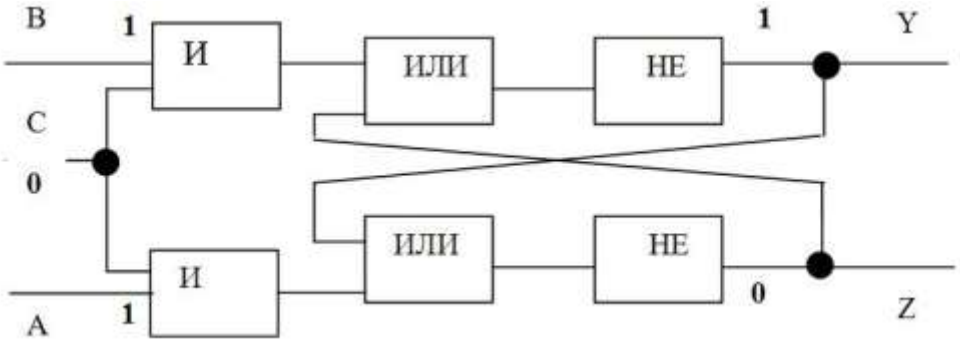
1) F=a and b and (not c or not d)
2) F=a and not c and (not b or not d)
3) F= (not a or b) and c and d
4) F=a and d and (not b or not c)
5) F=not b and (not c or a)

5. Укажите аналитическое представление логической функции F(a,b,c), которая для случаев F(0,0,0) и F(0,0,1) принимает значение "истина", а для остальных – "ложь". (and, or, not – знаки логических операций конъюнкции, дизъюнкции и отрицания соответственно.)

1) (not a and not b and c) or (not a and b and not c) or (not a and b and c) or (a and not b and not c)
2) not a and not b and c
3) (not a and not b and not c) or (not a and not b and c)
4) (not a and not b and not c) or (not a and b and not c) or (not a and b and c) or (a and not b and not c)
5) not a and not b and not c

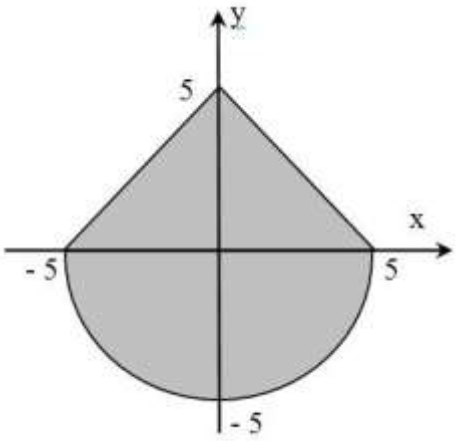
6. Начальное состояние схемы с памятью приведено на рисунке. Какие логические сигналы необходимо установить на входах A,B,C, чтобы на выходах Y и Z были получены логические сигналы: Y=0, Z=1 ?

1) A=0, B=1, C=1
2) A=0, B=1, C=0
3) A=1, B=0, C=1
4) A=0, B=0,C=0



7. Определите выражение, которое соответствует геометрическому месту точек, представленному на рисунке в виде заштрихованной области. (and, or - знаки логических операций конъюнкции и дизъюнкции соответственно.) Границы заштрихованной области входят в её состав.

1) (y <= x + 5) and (y >= - x - 5) and (25 >= x*x + y*y)
2) (y <= x + 5) and (y <= 5 - x) and (25 >= x*x + y*y)
3) (y <= 5 - x) and (y >= x - 5) and (25 >= x*x + y*y)
4) (y >= - 5 - x) and (y >= x - 5) and (25 >= x*x + y*y)



8. Определены четыре множества: A={0, 3}, B={3, 1,8,7}, C={8,11,6,3,21}, D={2,11,3,8,14}. Определить множество F =(A U B) ∩(C ∩ D), где ∩ - знак операции пересечения, а U – знак операции объединения множеств.

1) {3,8}
2) {5,7}
3) {3}
4) Пустое множество

9. Укажите синтаксически правильную формулу, записанную в ячейку электронной таблицы.

1) =ЕСЛИ(A1>10; 5; -5)
2) =ЕСЛИ(A1>10 ? 5: -5)
3) ЕСЛИ(ИЛИ(A1>10;B1<0);5;-5)
4) =ЕСЛИ(ИЛИ(A1>10,B1<0),5,-5)
5) =ЕСЛИ(ИЛИ(A1>10;B1<0);5;-5)

10. Ниже приведены программы на пяти языках программирования, реализующие один и тот же алгоритм. Выберите одну из программ и определите числа, которые будут выведены при выполнении программы. Примечание: не обращайте внимание на количество пробелов между числами.

1) 10 12 6
2) 15 15 7
3) 10 12 4
4) 15 15 8
5) 1 1 3 3 6

PYTHON

```
n = 5; m = 3
a=[[i+j+1 for i in range(m)] for j in range(n)]
p = 0
for j in range(m):
   x=0
   for i in range (p, n) :
      x = x + a[i][j]
   print (x, end=' ')
   p=p+2
```

PASCAL

```
program PR;
const m=3; n = 5;
var   a:array[1..n,1..m] of integer;
     i, j ,P, X : integer;
begin
    for i:=1 to n do
         for j:=1 to m do
            a[i, j] := i+j-1;
     P := 1;
     for j:=1 to m do
         begin
           X := 0;
           for i := P to n  do  X := X + a[i, j];
           write(X,' ');
        P := P + 2
       end
```

```
end.
```

C

```
#include <stdio.h>
#define N  5
#define M  3

void main() {
  int a[N][M], i, j, P, X;
  for(i=0; i< N; i++)
    for(j=0; j <M; j++)
          a[i][j] = i+ j+1;

  for(P=0, j=0;  j< M;  P=P+2,  j++)  {
     for(X=0, i=P;  i <N;  i++)  X = X + a[i][j];
     printf("%d ",X);
   }
}
```

C#

```
class Program
{
   static void Main(string[] args)
   {
      const int N = 5, M = 3;
      int[,] a = new int[N, M];
      int i, j, P, X;

      for (i = 0; i < N; i++)
         for (j = 0; j < M; j++)
            a[i, j] = i+j+1;

      for (P = 0, j = 0; j < M; P = P + 2, j++)
      {
         for (X = 0, i = P; i < N; i++)  X = X + a[i, j];
         System.Console.Write(X + " ");
      }
   }
}
```

BASIC

```
N = 5: M = 3
DIM A(N, M)
FOR I = 1 TO N
   FOR J = 1 TO M
```

```
        A(I, J) = I+J-1
    NEXT J
NEXT I
P = 1
FOR J = 1 TO M
    X = 0
    FOR I = P TO N
        X = X + A(I, J)
    NEXT I
    PRINT X," ";
    P = P + 2
NEXT J
```

Задания с вычислением и записью ответа

11. Какова разрядность кодирования стерео аудио файла длительностью звучания 16 секунд, с частотой дискретизации 240 Кгц? Размер файла - 7500 Кбайт.

Ответ: _____

12. Используя запросы, приведенные в таблице, поисковый сервер выполнил поиск страниц. За время выполнения всех запросов состояние сети Интернет не менялось. В языке запросов символ «|» обозначает логическую операцию «ИЛИ», а «&» - логическую операцию «И». В таблице приведено количество найденных по запросам страниц.

Запрос	**Найдено страниц (в сотнях тысяч)**
Пекин	250
Вена	170
Берлин	160
Пекин & Вена	140
Пекин & Берлин	130
Вена & Берлин	120
Пекин | Берлин | Вена	310

Какое количество страниц (в сотнях тысяч) будет найдено по запросу *Пекин & Вена & Берлин*?

Ответ: _____

13. Используя всевозможные комбинации букв A, B, C, D, E, была составлена таблица пятибуквенных слов. В этой таблице все слова упорядочены по алфавиту и пронумерованы. Начало таблицы приведено ниже.

1. AAAAA
2. AAAAB
3. AAAAC
4. AAAAD
5. AAAAE
6. AAABA

……

Запишите слово, которое в таблице стоит под номером **136.**

Ответ: _____

14. Сколько единичек содержится в двоичной записи значения выражения:

$$2^{2048} - 4^{512} - 1024$$

Ответ: _____

15. Ниже на пяти языках программирования записана рекурсивная функция F.
Все программы реализуют один и тот же алгоритм. Выберите одну из программ и определите, чему будет равна сумма всех чисел, напечатанных на экране при выполнении вызова F(4)?

Ответ: _____

PYTHON

```
def  F(n):
   if n > 0:
      if n % 2 == 1:
         F(n-2)
      else:
         F(n-1)

      print (n)
```

Basic

```
SUB F(n)
   IF n > 0 THEN
      IF n MOD 2 = 1 THEN
         F(n - 2)
      ELSE
```

```
        F(n - 1)
      END IF
      PRINT n
   END if
END SUB
```

Pascal

```
procedure F(n: integer);
begin
        if n >0 then
        begin
            if n mod 2 = 1 then
                F(n - 2)
            else
                F(n - 1);
            writeln(n)
        end
end;
```

C

```
void F(int n)
{
        if (n > 0) {
                if (n % 2 == 1) F(n - 2);
                else  F(n - 1);
                printf("%d", n);
        }
}
```

C#

```
static void F(int n)
{
   if (n > 0)
   {
      if (n % 2 == 1)  F(n - 2);
      else            F(n - 1);
      System.Console.Write(n);
   }
}
```

16. На столе лежит 32 листа бумаги: 4 листа желтого цвета, остальные – белого. Известно, что со стола взяли лист, и он оказался желтым. Каково количество информации в этом сообщении?

Ответ: _____

17. Представленные ниже программы реализуют один и тот же алгоритм. Выберите одну из пяти программ на знакомом Вам языке. Какое значение будет выведено при выполнении программы?

Ответ: _____

Python

```
a = []
n=5
for i in range(n):
    a.append([])
    for j in range(n):
        if (i % 2 == 0):
            a[i].append(i + 1)
        else:
            a[i].append(n - i)
s=0
for i in range(n):
    s = s + a[i][i] + a[i][n - i - 1]
print(s)
```

PASCAL

```
program PR;
const n = 5;
type mas=array[1..n,1..n] of integer;
var a:mas; i,j,s:integer;
begin
  for i:=1 to n do
    for j:=1 to n do
      if (i mod 2 = 1) then
        a[i,j]:= i
      else
        a[i,j]:=n-i+1;
  s:=0;
  for i:=1 to n do
      s: = s + a[i, i] + a[i, n - i+1];

  write(s)
end.
```

C

```
#include <stdio.h>
```

```
#define N  5
void main()  {
  int a[N][N], i, j, s;

  for(i=0; i< N; i++)
    for(j=0; j< M; j++)
      if(i%2==0)  a[i][j]=i+1;
      else            a[i][j]=N-i;

        for (i = 0, s = 0; i < N; i++)
            s = s + a[i][j]+ a[i][ N - i - 1];
      printf("%d",s);
}
```

C#

```
class Program
{
   static void Main()
   {
      int N = 5, i, j, s;
      int[,] a = new int[N, N];

      for (i = 0; i < N; i++)
         for (j = 0; j < N; j++)
            if (i % 2 == 0)  a[i, j] = i + 1;
            else                a[i, j] = N - i;

      for (i = 0, s = 0; i < N; i++)
         s = s + a[i, i] + a[i, N - i - 1];
      System.Console.Write(s);
   }
}
```

BASIC

```
n = 5
DIM a(n, n)
FOR i = 1 TO n
  FOR j = 1 TO n
    IF (i MOD 2 = 1) THEN
      a(i, j) = i
    ELSE
      a(i, j) = n - i + 1
    END IF
```

```
  NEXT j
NEXT i

s = 0
FOR i = 1 TO n
  s = s + a(i, i) + a(i, n - i+1)
NEXT i
PRINT s
```

18. Шестнадцатеричное число A = -1B0 представлено в формате с фиксированной точкой в дополнительном коде. Длина формата – 12 двоичных разрядов. Определите, какому шестнадцатеричному числу будет соответствовать этот код после инвертирования значений четных битов и логического сдвига вправо на один двоичный разряд. Номера бит отсчитываются справа налево, начиная с нуля. Младший бит – бит с номером 0, четный. Для записи шестнадцатеричных цифр со значением больше 9 используйте большие буквы латинского алфавита.

Ответ: _____

19. Определите максимально возможное количество строк на экране монитора, если известно, что объем видеопамяти, отображаемой на экран, 192000 байт, длина строки – 800 пикселей, каждый пиксель может отображать один из 16 цветов.

Ответ: _____

20. Алфавит состоит из 14 символов. Все символы кодируются одинаковым и минимально возможным количеством бит. Для хранения некоторого текста требуется 60 байтов памяти. Укажите, из скольких символов состоит текст.

Ответ: _____

Всероссийская олимпиада школьников
«Миссия выполнима. Твое призвание-финансист!»

ЗАДАНИЯ, РЕШЕНИЯ, КРИТЕРИИ
ЗАКЛЮЧИТЕЛЬНЫЙ (ОЧНЫЙ) ЭТАП
Информатика 10-11 класс, 2015/2016учебный год

ВАРИАНТ 1

Инструкция для участника олимпиады

Организационные указания

Продолжительность олимпиады – 120 минут (2 астрономических часа). Олимпиадное задание состоит из трех задач. Для каждой задачи указан ее вес. Участник олимпиады самостоятельно определяет последовательность выполнения задач. Участник олимпиады получает бумагу для черновика.

На одном из языков программирования – C/C++, C#, Basic, Pascal или Python – разработайте программы для решения перечисленных ниже задач.

Перед началом выполнения задания создайте на рабочем столе компьютера папку с именем следующего формата:

Олимпиада_*КодУчастника*

например, **Олимпиада_5034**

Внутри указанной папки Вы должны представить разработанные программы в формате выбранной среды программирования.

Первой строкой каждой программы должен быть комментарий в формате:

КодУчастника, **задача** *N*

например, для С-подобных языков:

// 5034, задача 1

Доступ к сети Интернету отключается. Исключается использование участником любых приемо-передающих устройств.

Сообщите члену комиссии об окончании ответа на задание и ждите его указаний.

Вычисление итоговой оценки

Каждая программа, разработанная участником олимпиады, оценивается членом жюри, исходя из 100 баллов. Если программа содержит недоработки или ошибки, то оценка снижается.

Каждую программу проверяет несколько членов жюри. Итоговая оценка $И_i$, выставляемая i-тым членом жюри, получается как сумма оценок, умноженных на соответствующий весовой коэффициент (Б*К). Итоговая оценка ИТОГО, выставляемая участнику всеми членами жюри, вычисляется как средняя арифметическая величина. Таким образом:

$$И_i = Б_1 * К_1 + Б_2 * К_2 + Б_3 * К_3 ,$$

где:

Б – оценка за программу по 100-бальной системе;

К – весовой коэффициент. $К_1 + К_2 + К_3 = 1.0$

$К_1 = 0,25$

$К_2 = 0,40$

$К_3 = 0,35$

Итоговая оценка выставляется по следующей формуле:

$$ИТОГО = (И_1 + И_{2+} \ldots И_n)/n$$

где: n – количество членов жюри, проверяющих программы.

$И_i$ – оценка, выставленная i-тым членом жюри.

Критерии оценивания программы

Поскольку разработка программы является творческим процессом, то и оценивание программы, в значительной степени, носит субъективный характер. Тем не менее, оценки, выставляемые разными членами жюри, как правило, имеют небольшой разброс, а оценивание несколькими членами жюри позволяет получить итоговую оценку, близкую к истинной.

Каждая программа оценивается по 100-бальной шкале в соответствии с показателями, перечисленными в следующей таблице:

Показатели оценивания	**Балл ($ПР_{100}$)**
1. Программа полностью соответствует условию, выполняет поставленную задачу и ни при каких обстоятельствах не завершается аварийно. 2. Пользовательский интерфейс эргономичен и интуитивно понятен. 3. Алгоритмы обработки данных эффективны и рациональны. 4. Обнаружены несущественные недостатки.	86-100
Не выполнены требования на оценку «отлично», при этом выявлены один или несколько недостатков: 1. Программа в основном соответствует условию задачи. Допущены несущественные отклонения от условия. 2. Программа завершается аварийно только при вводе некорректных данных или выполнении второстепенных функций. 3. Пользовательский интерфейс недостаточно эргономичен и интуитивно непонятен. 4. Алгоритмы обработки данных неэффективны. Существует более простой способ решения задачи.	70-85
Не выполнены требования на оценку «хорошо», при этом: 1. Программа имеет отклонения от условия. Задача решена частично. 2. Отдельные фрагменты не отлажены до конца, но в целом близки к правильному алгоритму.	50-69
1. Работу не удается идентифицировать или работа не найдена. 2. Программа имеет синтаксические ошибки. 3. Программа в целом не соответствует условию.	0-49

Участник олимпиады должен обратить внимание на следующие показатели качества программ:

- Отсутствие синтаксических ошибок;
- Отсутствие аварийных завершений программы;
- Соответствие программы условию задачи;
- Полнота отлаженности программы, вывод правильного ответа для любых допустимых исходных данных;
- Качество интерфейса;
- Эффективность и рациональность алгоритмов;
- Контроль вводимых с клавиатуры исходных данных;
- Читабельность программы;
- Наличие комментариев;
- Возможность повторного выполнения программы для новых исходных данных.

За неработающую программу член жюри может выставить некоторые баллы, если в программе частично реализован алгоритм, близкий к правильному.

Если отсутствует контроль вводимых с клавиатуры исходных данных, то 100 бальная оценка Б уменьшается до 8 баллов. За отсутствие комментариев оценка уменьшается на 2 балла. За плохую читабельность оценка уменьшается на 2 балла. За отсутствие повторения программы для новых исходных данных оценка уменьшается на 2 балла.

Если программа имеет синтаксические ошибки, оценка уменьшается не менее чем на 50 баллов.

Если программа содержит нерациональные и неэффективные, но работающие алгоритмы, оценка может быть снижена до 20 баллов.

Если программа для отдельных комбинаций исходных данных завершается аварийно, а для других – нет, оценка может быть снижена до 40 баллов.

Неполное соответствие программы условию задачи, неполная отлаженность программы, невысокое качество интерфейса и т.д. приводят к снижению оценки на величину, определяемую степенью недоработки программы.

Критерии определения победителей и призеров заключительного этапа олимпиады

Участник олимпиады, набравший наибольшее количество баллов, является победителем олимпиады. Следующие по количеству баллов участники получают соответственно второе и третье место. Участник участвует в конкурсе, если он набрал больше 49 баллов.

Задача 1. Вес 25 баллов

Отдельная пара носков стоит 105 руб., связка из 12 пар стоит 1025 руб., а коробка из 12 связок – 11400 руб. Составить программу, которая по введенному с клавиатуры числу N пар носков, которые должен купить покупатель, вычисляет, сколько необходимо купить коробок, связок и отдельных пар носков, чтобы стоимость покупки была минимальной.

В программе необходимо:

1. Найти состав оптимальной покупки без излишков, то есть должно быть куплено ровно N пар носков по минимальной цене.

2. Попытаться выполнить оптимизацию покупки так, чтобы стоимость покупки стала еще меньше, а общее количество покупаемых пар носков при этом стало больше введенного (>N).

3. Сравнить варианты 1 и 2 и вывести на экран полученную покупателем выгоду по стоимости покупки и общему количеству пар носков.

Примерное решение задачи 1 на языке C#:

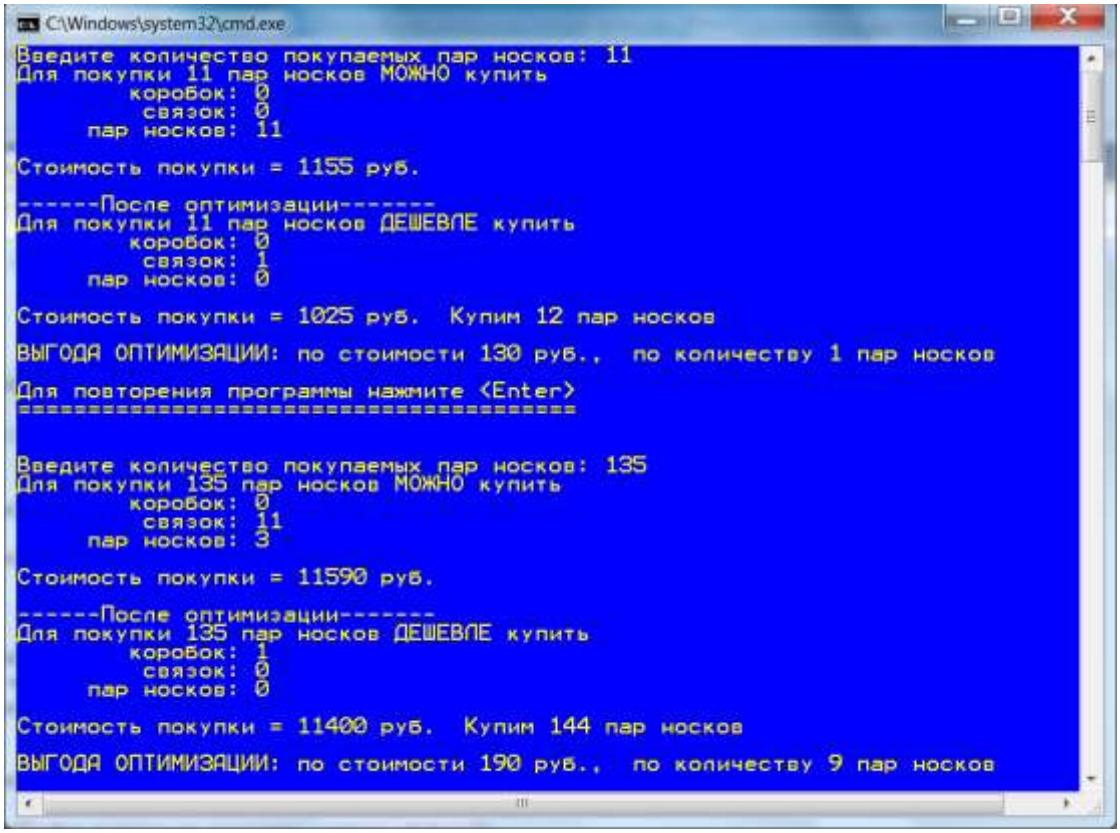
Особенность решения задачи состоит в том, что может быть сформирована оптимальная покупка без излишков и более дешевая покупка, но с излишками. Например, вместо 11 пар носков дешевле купить связку или вместо 2 пар и 11 связок лучше купить коробку.

```
using System;
class Program
{
   static void Main(string[] args)
   {
      int n1,     // Число коробок.
          n2,     // Число связок.
          n3,     // Число пар.
          n,      // Покупаемое число пар.
          opt,    // Оптимальное количество пар носков.
          m,
          s1, s2;     // Стоимость покупки.
      do
      {

         while (true)
         {
            Console.Write("Введите количество покупаемых пар носков: ");
            if (int.TryParse(Console.ReadLine(), out n)) break;
            Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
         }

         // Вычисляем оптимальную покупку для n пар носков.
         n1 = n / (12 * 12);     // Количество коробок.
         m = n % 144;            // Остаток.
         n2 = m / 12;            // Количество связок.
         n3 = m % 12;            // Количество пар.

         Console.WriteLine("Для покупки {0} пар носков МОЖНО купить" +
                           "\n       коробок: {1} \n       связок: {2} \n     пар носков: {3}",
                           n, n1, n2, n3);
         s1 = n3 * 105 + n2 * 1025 + n1 * 11400;
         Console.WriteLine("\nСтоимость покупки = {0} руб.", s1);

         // Оптимизируем покупку для количества пар > n.
         if (n3 * 105 > 1025)
         {
            n2++;       // Вместо 10 или 11 пар носков дешевле купить связку.
            n3 = 0;
         }

         if (n2 * 1025 + n3 * 105 > 11400)
         {
            n1++;       // Вместо 2 пар и 11 связок  дешевле купить упаковку.
            n2 = n3 = 0;
         }
         Console.WriteLine("\n------После оптимизации-------");

         Console.WriteLine("Для покупки {0} пар носков ДЕШЕВЛЕ купить" +
                           "\n       коробок: {1} \n       связок: {2} \n     пар носков: {3}",
                           n, n1, n2, n3);

         s2 = n3 * 105 + n2 * 1025 + n1 * 11400;
         opt = n1 * 144 + n2 * 12 + n3;
         Console.WriteLine("\nСтоимость покупки = {0} руб.  Купим {1} пар носков",
         s2, opt);
```

```
        Console.WriteLine("\nВЫГОДА ОПТИМИЗАЦИИ: по стоимости {0} руб.,  по количеству {1} пар носков", s1 - s2, opt - n);

        Console.WriteLine("\nДля повторения программы нажмите <Enter>");
        Console.WriteLine("=======================================\n\n");
      } while (Console.ReadKey(true).Key == ConsoleKey.Enter);

   }
}
```

Операторы, вычисляющие оптимальный состав покупки, выделены серым фоном.

Вывод:



Задача 2. Вес 40 баллов

Создать квадратную матрицу **A** размером **N**x**N** и заполнить ее целыми однозначными случайными числами. Значение **N** должно вводиться с клавиатуры.

Вычислить среднее арифметическое значение элементов матрицы, расположенных в треугольнике, образованном диагоналями и правой границей. Элементы сторон треугольника входят в его состав. Вывести на экран матрицу и вычисленное среднее арифметическое значение.

Создать двумерный массив **B**, состоящий из **N** строк. В эти строки из соответствующих строк матрицы **A** скопировать элементы, большие найденного среднего арифметического значения.

В каждой строке массива **B** найти минимальный элемент. Сохранить эти элементы в новом векторе. Если в строке несколько минимальных элементов, то сохранять только один. Если строка отсутствует, вывести символы **nr.**

Вывести на экран массив и вектор минимальных элементов.

Графическая иллюстрация этапов решения задачи для матрицы размером 6x6 приведена на следующем рисунке.

6	8	6	1	8	**4**
5	9	0	5	**8**	**7**
3	3	8	**6**	**0**	**5**
1	8	0	**0**	**8**	**9**
3	4	5	2	**2**	**7**
4	9	1	4	5	**4**

6	8	6	8
9	8	7	
8	6		
8	8	9	
7			
9			

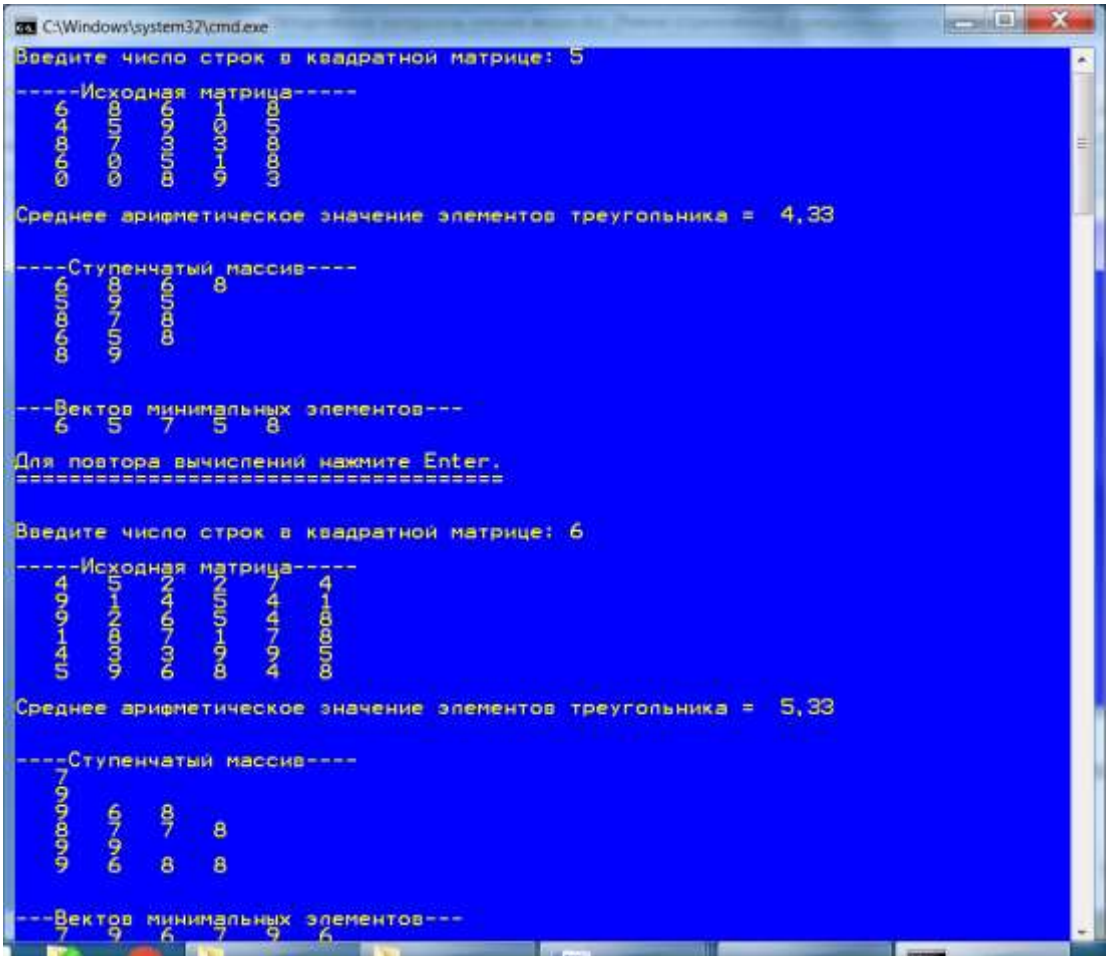
6	7	6	8	7	9

Примерное решение задачи 2 на языке C#:

```
using System;
class Program
{
   public static void Main()
   {
      Random rnd = new Random(17);         // Объект - генератор чисел

      do
      {
         int n;                  // Количество строк
         int[,] a;               // Ссылка на формируемую матрицу
         int[][] b;              // Ссылка на рваный массив
         int[] s;                        // Ссылка на вектор минимальных элементов.
         int[] k;                // Ссылка на массив с длинами строк.
         double sr;              // Среднее арифметическое значение элементов треугольника.
         int i, j, m;            // Номера строк и столбцов.

         Console.Write("Введите число строк в квадратной матрице: ");
         n = int.Parse(Console.ReadLine());

         a = new int[n, n];
         k = new int[n];
         s = new int[n];
         b = new int[n][];


         // Инициализация матрицы.
         for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
               a[i, j] = rnd.Next(0, 10);

         // Вывод матрицы.
         Console.WriteLine("\n-----Исходная матрица-----");
         for (i = 0; i < n; i++, Console.WriteLine())
            for (j = 0; j < n; j++)
               Console.Write("{0,4}", a[i, j]);

         //Вычисление среднего арифметического значения элементов треугольника.
         //      Верхняя половинка треугольника:
         sr = 0;
         for (i = 0; i < n / 2 + n % 2; i++)     // n%2 - +1 для нечетного числа строк
            for (j = n - i - 1; j < n; j++)
               sr += a[i, j];

         //      Нижняя половинка:
         for (i = n / 2 + n % 2; i < n; i++)
```

```
            for (j = i; j < n; j++)
                sr += a[i, j];

        sr = sr / ((n * (n - 2) + n % 2) / 4 + n);          // Среднее арифметическое.

        Console.WriteLine(
              "\nСреднее арифметическое значение элементов треугольника = {0: 0.00}", sr);

        // Подсчитаем длину каждой строки ступенчатого массива.
        for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
                if (a[i, j] > sr)
                    k[i]++;

        // Создадим строки ступенчатого массива.
        for (i = 0; i < n; i++)
            b[i] = new int[k[i]];

        // Скопируем элементы в строки ступенчатого массива.
        for (i = 0; i < n; i++)
            for (j = 0, m = 0; j < n; j++)
                if (a[i, j] > sr)
                {
                    b[i][m] = a[i, j];
                    m++;
                }

        // Вывод массива.
        Console.WriteLine("\n\n----Ступенчатый массив----");

        for (i = 0; i < n; i++, Console.WriteLine())
            for (j = 0; j < b[i].Length; j++)
                Console.Write("{0,4}", b[i][j]);

        // Найдем мин. элементы.
        for (i = 0; i < n; i++)
        {
            if (b[i].Length > 0)
                s[i] = b[i][0];
            for (j = 0; j < b[i].Length; j++)
                if (s[i] > b[i][j])
                    s[i] = b[i][j];
        }

        // Вывод вектора минимальных элементов.
        Console.WriteLine("\n\n---Вектов минимальных элементов---");

        for (i = 0; i < n; i++)
            if (b[i].Length > 0)
                Console.Write("{0,4}", s[i]);
            else
                Console.Write("nr");

        Console.WriteLine("\n\nДля повтора вычислений нажмите Enter.");
        Console.WriteLine("==========================================\n\n");
    } while (Console.ReadKey(false).Key == ConsoleKey.Enter);
  }
}
```

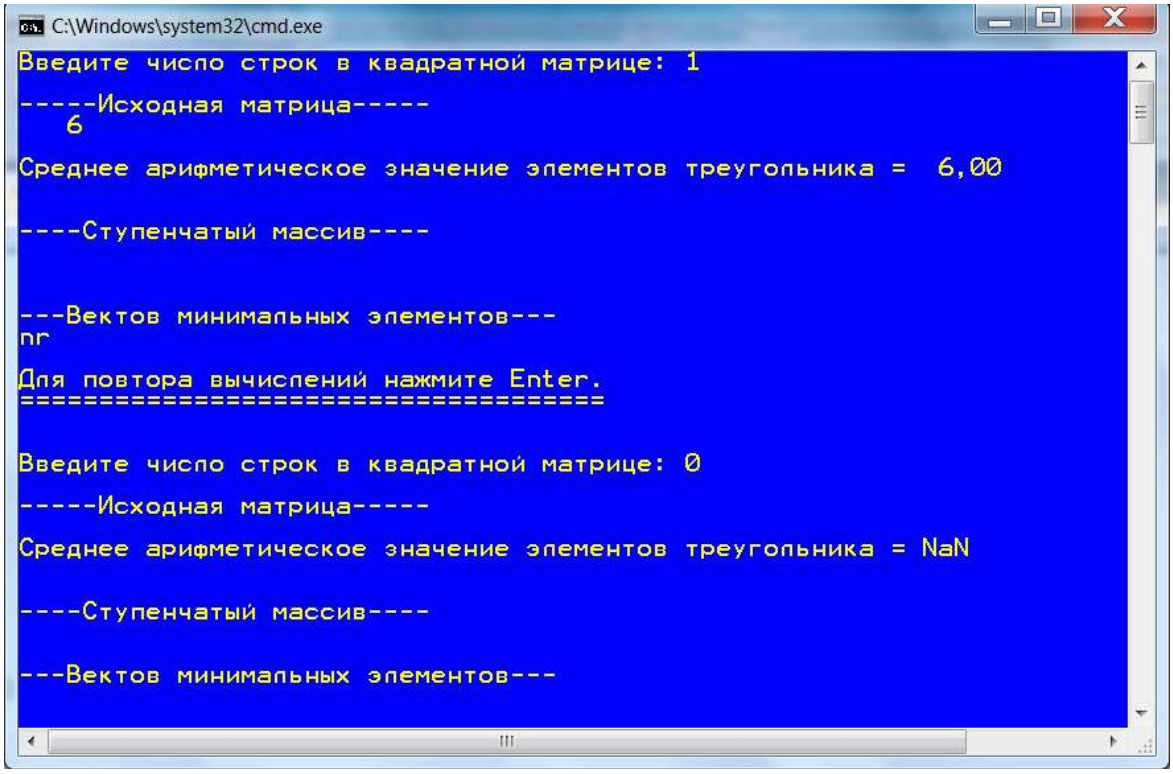
Вывод:



Примечание. Обратите внимание, как вычисляется количество элементов в треугольнике. Так как массив **B** должен быть массивом массивов, то есть ступенчатым (рваным) массивом, то в этом массиве отдельные строки могут отсутствовать, при этом минимальное значение для этих строк выводиться не должно. В распечатке окна, ниже, для таких строк печатается **nr** – нет строки.

Вывод массивов должен соответствовать табличной форме.

Программа должна работать как для четного значения N, так и для нечетного, как для N=0, так и для N=1. Для N=1 ступенчатый массив всегда пустой, поэтому печатается **nr**. Например:



Задача 3. Вес 35 баллов

Имеется некоторая сумма денег и набор купюр достоинством A_1, …, A_n, которые вводятся с клавиатуры. Необходимо найти все возможные способы разменять сумму при помощи этих купюр. Каждую указанную купюру можно использовать только один раз, однако в наборе та или иная купюра может повторяться несколько раз. Количество купюр в наборе может быть произвольным, но не должно превышать максимальное значение для типа int (Integer).

Для получения максимальной оценки решение должно быть основано на реализации *рекурсивного* алгоритма.

Примерные решения задачи 3 на языке C#

Основа алгоритма состоит в создании всех сочетаний купюр из набора и проверки равенства исходной суммы сумме купюр, соответствующей очередному сочетанию.

Способ 1. Используем рекурсивную функцию. Макс. оценка – 35 баллов.

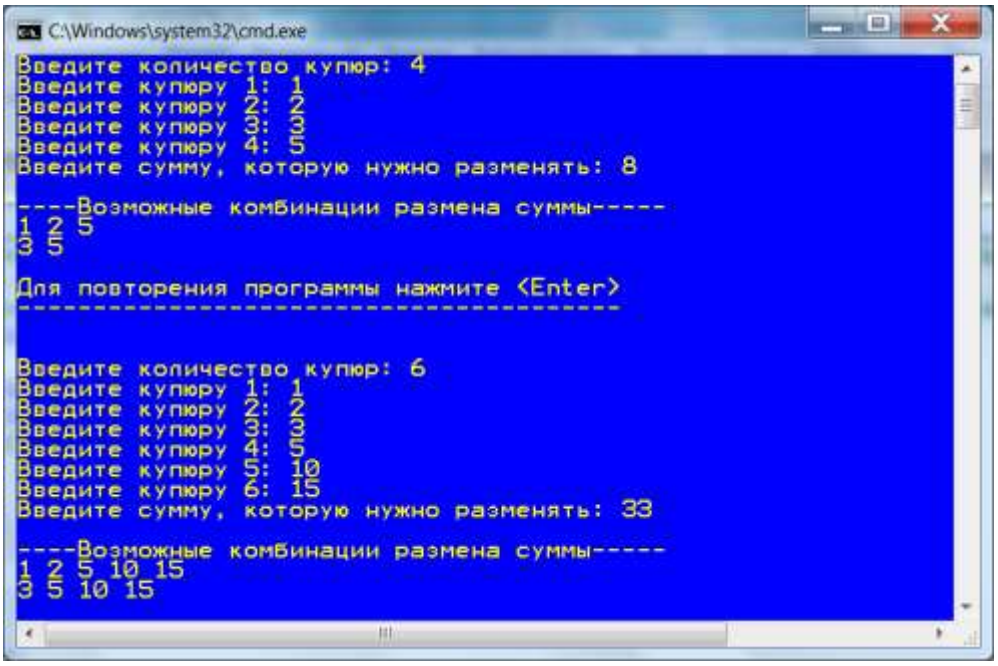
С помощью рекурсии формируются все возможные комбинации из введенных купюр. Фрагменты программы, осуществляющие основные вычисления, показаны на сером фоне.

```
using System;

class Program
{
   // Глобальные переменные (поля).
   static int s ;                // Исходная сумма
   static int[] a ;              // Набор купюр
   static int n ;                // Число купюр
   static int[] сочетание ;      // 1 - купюра включается в накапливаемую сумму, 0 - нет.
   static bool found = false;

   static void Main()
   {
```

```
    do
    {
      // Ввод исходных данных.
      while (true)
       {
          Console.Write("Введите количество купюр: ");
          if (int.TryParse(Console.ReadLine(), out n)) break;
          Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
       }

       a =   new int[n];
       сочетание = new int[n];

       for (int i = 0; i < n; i++)
       {
          Console.Write("Введите купюру {0}: ", i+1);
          if (!int.TryParse(Console.ReadLine(), out a[i]))
          {
             Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
             i--;
          }
       }


       while (true)
       {
          Console.Write("Введите сумму, которую нужно разменять: ");
          if (int.TryParse(Console.ReadLine(), out s)) break;
          Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
       }


       Console.WriteLine("\n----Возможные комбинации размена суммы-----");
       Размен(-1, 0);
       if (!found) Console.WriteLine("Разменять невозможно!");



       Console.WriteLine("\nДля повторения программы нажмите <Enter>");
       Console.WriteLine("----------------------------------------\n\n");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);

}


//-------------------------------------------------------------
// Размен - рекурсивная функция (метод) размена суммы s.
// Функция проверяет, равна ли сумма взятых купюр исходной сумме.
// k – число уже использовавшихся купюр, s1 – набранная сумма
// Возвращаемое значение формируется в глобальной переменной:
//           true - разменяли, false - размен не возможен.
static void Размен(int k, int s1)
{
   int i;
   if (s == s1)
   {  // Вывод найденного решения.
      found = true;
      for (i = 0; i < n; i++)
         if (сочетание[i] == 1)
            Console.Write(a[i] + " ");

      Console.WriteLine();
      return;
   }
```

```
        for (i = k+1; i < n; i++)  // Цикл по всем еще не проверенным купюрам
        {
          сочетание[i] = 1;       // Включим в накапливаемую сумму следующую купюру.
          //Добавляем к сумме s1 купюру a[i] и проверяем новую сумму на равенство исходной.
          Размен(i, s1 + a[i]);
          сочетание[i] = 0;       // Исключим i-ю купюру из сочетания, чтобы взять следующую.
        }
    }
}
```

Вывод:



Способ 2. Макс. оценка – 30 баллов. Задача решается для любого числа купюр с помощью одного цикла, который генерирует числа от 0 до 2^n-1. Эти числа соответствуют всем возможным комбинациям купюр. В двоичном числе: 1 - соответствующая купюра входит в комбинацию, 0 - не входит. Каждая двоичная цифра (0 или 1) в двоичном представлении числа соотносится с соответствующей купюрой в наборе. Разряды выделяются, проверяются и удаляются сдвигом.

На каждом шаге:

- выделяется последний разряд числа, если он равен 1, накапливается сумма.
- далее следует сдвиг числа на 1 разряд вправо.

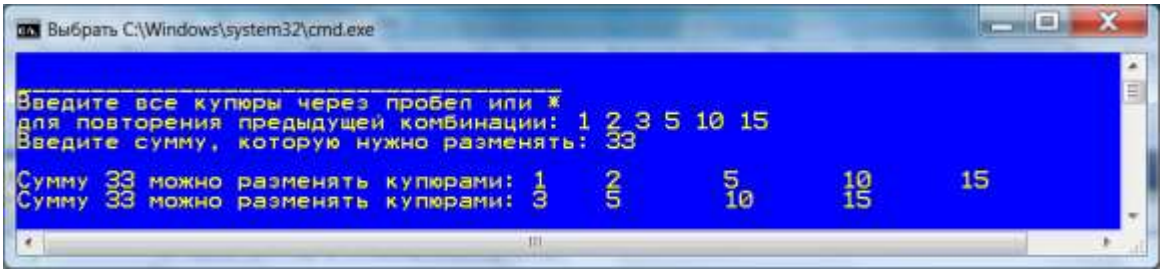
В представленном решении большое место занимает ввод исходных данных: купюры вводятся одной строкой, их количество вычисляется программой; можно менять только сумму, оставляя набор купюр прежним. Основные фрагменты программы отмечены серым фоном.

```
using System;
using System.Collections.Generic;

class Program
{
   static void Main(string[] args)
   {
      List<int> набор = new List<int>();          // Список купюр.
      string купюрыStrOld = "1 2 3 5 10 15";

      do
```

```
{
    int summa;                          // Размениваемая сумма.
    bool естьРазмен = false,            // false - сумму нельзя разменять.
       notError = true;                 // Признак корректного ввода.

    // Ввод исходных данных.
    while (notError)
    {
       Console.WriteLine("\n_____");

       Console.Write("Введите все купюры через пробел или *\n" +
                     "для повторения предыдущей комбинации: ");
       string купюрыStr = Console.ReadLine();
       if (купюрыStr == "*")
       {
          Console.WriteLine("Предыдущая комбинация купюр: " + купюрыStrOld);
          break;
       }
       набор.Clear();
       купюрыStrOld = купюрыStr;

       char[] sep = { ' ' };
       string[] купюры = купюрыStr.Split(sep, StringSplitOptions.RemoveEmptyEntries);

       for (int i = 0; i < купюры.Length; i++)
       {
          int s;
          if (!int.TryParse(купюры[i], out s))
          {
             notError = true;
             Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
             break;
          }
          else
          {
             набор.Add(s);
             notError = false;
          }
       }
    }


    while (true)                        // Бесконечный цикл
    {
       Console.Write("Введите сумму, которую нужно разменять: ");

       if (int.TryParse(Console.ReadLine(), out summa)) break;
       Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
    }
    Console.WriteLine();




    // Решение задачи.
    int n = набор.Count;                // Количество купюр.

    // Получим все числа от 0000...0000 до 1111...1111.   1-есть купюра, 0 - нет.
    int k = (int)Math.Pow(2, n);   //Максимальное количество комбинаций (1 000 ... 00).
    for (int num = 1; num < k; num++)
       if (Проверить(num, набор, summa)) естьРазмен = true;


    if (!естьРазмен)
```

```
            Console.WriteLine("\nВведенную сумму нельзя разменять перечисленными купюрами!");



            Console.WriteLine("\nДля повторения программы нажмите <Enter>");
        } while (Console.ReadKey(true).Key == ConsoleKey.Enter);
    }

    //-----------------------------------------------------------------
    // Метод находит сумму купюр, соответствующих разрядам переданного числа num.
    // набор - список купюр, summa - сумма.
    private static bool Проверить(int num, List<int> набор, int summa)
    {
        int s = 0;              // Сумма, соответствующая переданному числу num.
        string размен = "";


        // Определим сумму по единичным битам числа.
        for (int i = 0; i < набор.Count; i++)
        {
            if (num == 0) break;
            if ((num & 1) == 1)
            {
                s += набор[i];
                размен += набор[i] + "\t";
            }

            if (s > summa)  return false;

            num = num >> 1;
        }


        if (s == summa)
        {
            Console.WriteLine("Сумму {0} можно разменять купюрами: {1}", summa, размен);
            return true;
        }
        else
            return false;
    }
}
```

Вывод:



ВАРИАНТ 2

Задача 1. Вес 35 баллов

Строители укладывают дорогу бетонными плитами, одна к другой без промежутков. Длина дороги L метров. На складе имеется n пронумерованных плит: от 1 до n. Все бетонные плиты имеют одинаковую ширину, равную ширине дороги, поэтому с подбором плит по ширине дороги

проблемы отсутствуют. Однако длина каждой плиты может быть произвольной: на складе есть как одинаковые, так и разные по длине плиты. Длина любой плиты кратна дециметру.

Необходимо найти все возможные способы укладки дороги плитами, так чтобы их общая длина составляла ровно L метров.

Длину дороги, количество плит на складе и их длины вводятся с клавиатуры.

Количество плит может быть ограничено здравым смыслом, например не больше максимального значения переменной типа int (Integer).

Решение с помощью рекурсии – 35 баллов.

Решение без рекурсии – 30 баллов.

Примерные решения задачи 1 на языке C#

Основа алгоритма состоит в создании всех сочетаний складских плит, представленных их длинами, и проверки для каждого сочетания равенства длины дороги общей длине используемых плит.

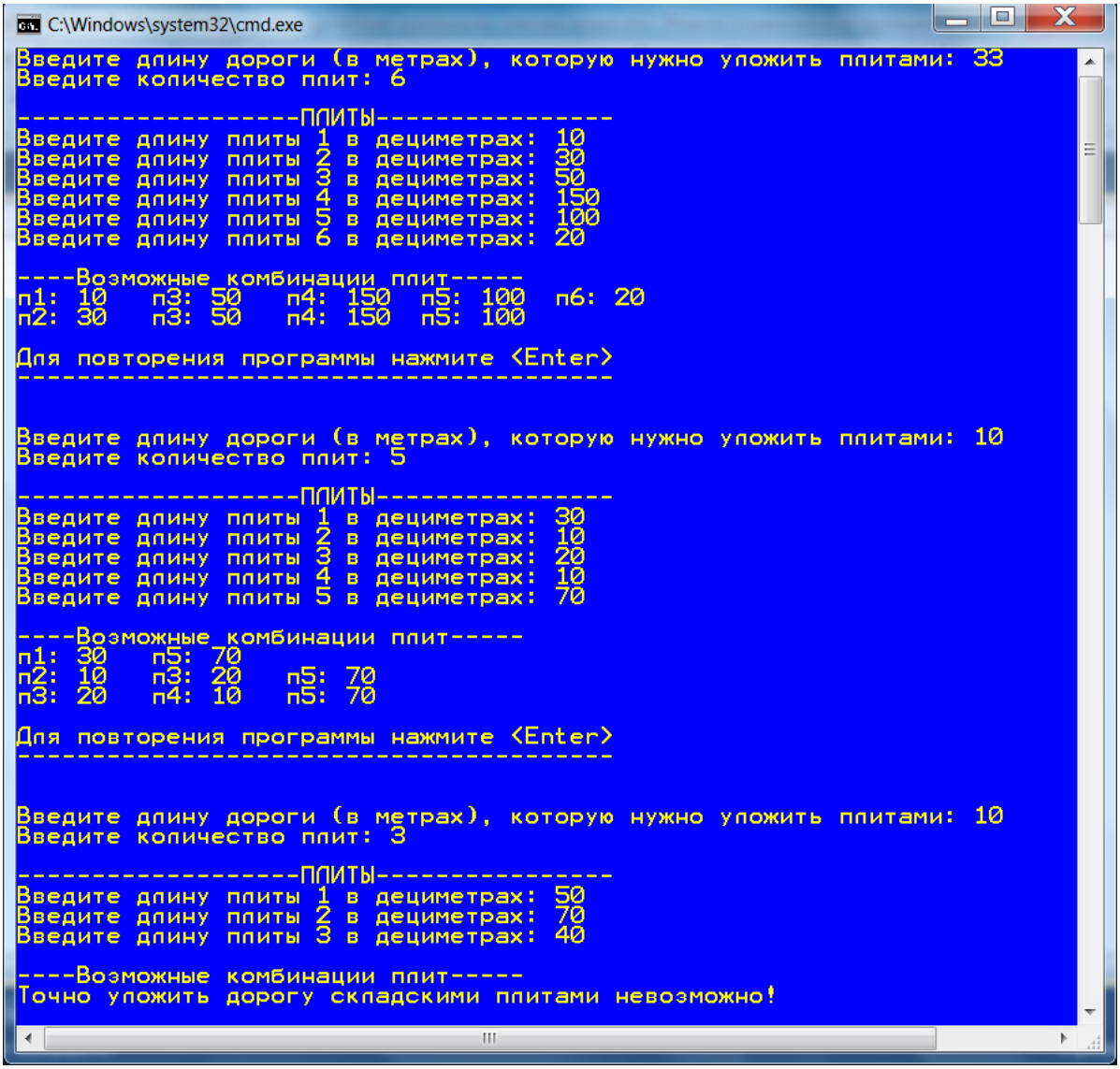
Способ 1. *Используем рекурсивную функцию. Макс. оценка – 35 баллов.*

С помощью рекурсии формируются все возможные комбинации из введенных длин. Фрагменты программы, осуществляющие основные вычисления, показаны на сером фоне.

```
using System;

class Program
{
   // Глобальные переменные (поля).
   static int    Lroad;              // Длина дороги
   static int[] L;                   // Набор плит (длин)
   static int    n;                  // Число плит
   static int[] сочетание;           // 1 - плита длиной L[i] включается в укладку дороги, 0 - нет.
   static bool found;

   static void Main()
   {
     do
     {
        // Ввод исходных данных.

       while (true)
       {
         Console.Write("Введите длину дороги (в метрах), которую нужно уложить плитами: ");
         if (int.TryParse(Console.ReadLine(), out Lroad)) break;
         Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
       }
       Lroad *= 10;

       while (true)
       {
           Console.Write("Введите количество плит: ");
           if (int.TryParse(Console.ReadLine(), out n)) break;
           Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
       }

         L = new int[n];
         сочетание = new int[n];
         found = false;

         Console.WriteLine("\n-------------------ПЛИТЫ----------------");

         for (int i = 0; i < n; i++)
```

```
        {
            Console.Write("Введите длину плиты {0} в дециметрах: ", i + 1);
            if (!int.TryParse(Console.ReadLine(), out L[i]))
                Console.WriteLine("\nВы ошиблись при вводе. Повторите!", i--);
        }


        Console.WriteLine("\n----Возможные комбинации плит-----");
        Укладка(-1, 0);
        if (!found) Console.WriteLine ("Точно уложить дорогу складскими плитами невозможно!");


        Console.WriteLine("\nДля повторения программы нажмите <Enter>");
        Console.WriteLine("----------------------------------------\n\n");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);

}


//-------------------------------------------------------------------------------------------------
// Укладка - рекурсивная функция (метод) укладки дороги плитами.
// Функция проверяет, равна ли точно длина всех используемых для укладки плит длине дороги.
// k – число уже использовавшихся плит со склада, sl – набранная сумма длин.
// Возвращаемое значение формируется в глобальной переменной:
//       true - уложили точно, false - плиты не подходят для укладки дороги.
static void Укладка(int k, int sl)
{
    int i;
    if (Lroad == sl)
    {  // Вывод найденного решения.
        found = true;
        for (i = 0; i < n; i++)
            if (сочетание[i] == 1)
                Console.Write("п{0}: {1, -5}", i+1, L[i]);

        Console.WriteLine();
        return;
    }

    // Цикл по всем еще не проверенным плитам.
    for (i = k + 1; i < n; i++)
    {
        сочетание[i] = 1;          // Включим в накапливаемую длину следующую плиту.
        //Добавляем к сумме sl плиту L[i] и проверяем новую сумму длин на равенство длине дороги.
        Укладка(i, sl + L[i]);
        сочетание[i] = 0;       // Исключим i-ю плиту из сочетания, чтобы взять следующую.
    }
}
}
```

Вывод:



Способ 2. *Макс. оценка – 30 баллов.*

Задача решается для любого набора плит с помощью одного цикла, который генерирует числа от 0 до 2^n-1. Эти числа соответствуют всем возможным комбинациям плит, хранящихся на складе. В двоичном числе: 1 - соответствующая плита входит в комбинацию, 0 - не входит. Каждая двоичная цифра (0 или 1) в двоичном представлении числа соотносится с соответствующей плитой (длиной) на складе. Разряды выделяются, проверяются и затем удаляются сдвигом.

На каждом шаге:

- выделяется последний разряд числа и, если он равен 1, накапливается сумма длин.
- далее следует сдвиг числа на 1 разряд вправо.

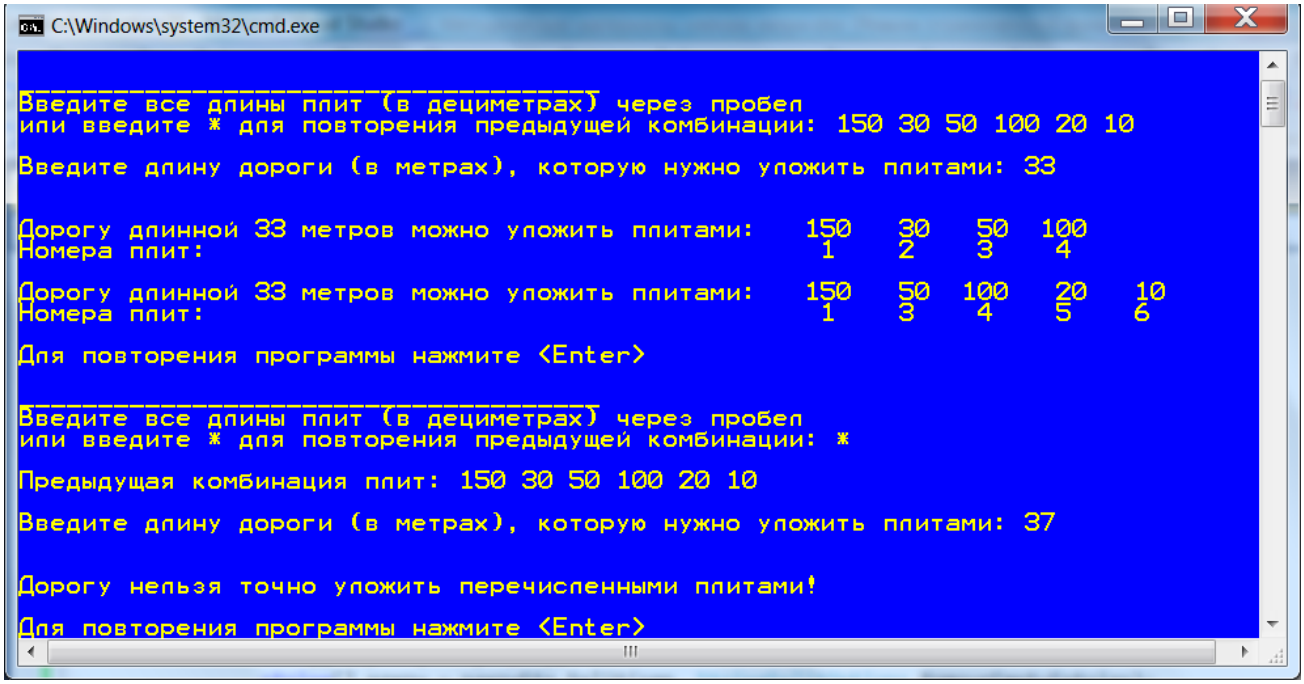
В представленном решении большое место занимает ввод исходных данных: длины плит вводятся одной строкой, их количество вычисляется программой; можно менять только длину дороги, оставляя набор плит прежним. Основные фрагменты программы отмечены серым фоном.

```
using System;
using System.Collections.Generic;

class Program
{
   static void Main(string[] args)
   {
```

```
List<int> набор = new List<int>();         // Список плит (длин).
string плитыStrOld = "10 20 30 50 100 150";

do
{
   int Lroad;                              // Длина дороги.
   bool естьУкладка = false;               // false - дорогу нельзя точно уложить плитами.
   bool notError = true;                   // Признак корректного ввода.

   // Ввод исходных данных.
   while (notError)
   {
      Console.WriteLine("\n_____");

      Console.Write("Введите все длины плит (в дециметрах) через пробел \n" +
                    "или введите * для повторения предыдущей комбинации: ");
      string плитыStr = Console.ReadLine();
      if (плитыStr == "*")
      {
         Console.WriteLine("\nПредыдущая комбинация плит: " + плитыStrOld);
         плитыStr = плитыStrOld;
      }
      набор.Clear();
      плитыStrOld = плитыStr;

      char[] sep = { ' ', ',' };
      string[] плиты = плитыStr.Split(sep, StringSplitOptions.RemoveEmptyEntries);

      for (int i = 0; i < плиты.Length; i++)
      {
         int s;
         if (!int.TryParse(плиты[i], out s))
         {
            notError = true;
            Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
            break;
         }
         else
         {
            набор.Add(s);
            notError = false;
         }
      }
   }


   while (true)                            // Бесконечный цикл
   {
      Console.Write("\nВведите длину дороги (в метрах), которую нужно уложить плитами: ");

      if (int.TryParse(Console.ReadLine(), out Lroad)) break;
      Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
   }
   Console.WriteLine();

   Lroad *= 10;       // В дециметрах.


   // Решение задачи.
   int n = набор.Count;                // Количество плит.

   // Получим все числа от 0000...0000 до 1111...1111.   1- плита включена, 0 - нет.
   int k = (int)Math.Pow(2, n);   //Максимальное количество комбинаций (1 000 ... 00).
   for (int num = 1; num < k; num++)
```

```
            if (Проверить(num, набор, Lroad))   естьУкладка = true;


        if (!естьУкладка)
            Console.WriteLine("\nДорогу нельзя точно уложить перечисленными плитами!");



        Console.WriteLine("\nДля повторения программы нажмите <Enter>");
    } while (Console.ReadKey(true).Key == ConsoleKey.Enter);
}

//-----------------------------------------------------------------
// Метод находит сумму длин плит, соответствующих разрядам переданного числа num.
// набор - список плит (длин), длина - сумма длин.
private static bool Проверить(int num, List<int> набор, int длина)
{
    int s = 0;              // Сумма, соответствующая переданному числу num.
    string укладка = "";
    string ind = "";


    // Определим сумму по единичным битам числа.
    for (int i = 0; i < набор.Count; i++)
    {
        if (num == 0) break;
        if ((num & 1) == 1)
        {
            s += набор[i];
            укладка += string.Format("{0, 5}", набор[i]);
            ind += string.Format("{0, 5}", i+1);
        }

        if (s > длина) return false;

        num = num >> 1;
    }


    if (s == длина)
    {
        Console.WriteLine("\nДорогу длинной {0} метров можно уложить плитами: {1}",
                          длина / 10, укладка);
        Console.WriteLine("Номера плит:                          {0}", ind);
        return true;
    }
    else
        return false;
}
}
```

Вывод:



Задача 2. Вес 40 баллов

Создать квадратную матрицу **M** размером **N**x**N** и заполнить ее целыми однозначными случайными числами. Значение **N** должно вводиться с клавиатуры.

Вычислить среднее арифметическое значение элементов матрицы, расположенных в треугольнике, образованном диагоналями и нижней границей. Элементы сторон треугольника входят в его состав. Вывести на экран матрицу и вычисленное среднее арифметическое значение.

Создать двумерный массив **B**, состоящий из **N** строк. В эти строки из соответствующих полных *столбцов* матрицы **M** (первый столбец в первую строку, второй – во вторую и т.д.) скопировать элементы, меньшие найденного среднего арифметического значения. Вывести массив **B** на экран. Если строка в массиве **B** пустая (отсутствует), вывести на экран символы **nr.**

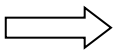
В каждой строке массива **B** найти максимальный элемент. Сохранить эти элементы в новом векторе **V** длиной **N** элементов. Если в строке несколько максимальных элементов, то сохранить только один. Вывести вектор **V** на экран. Если строка в массиве **B** пустая (отсутствует), то вместо соответствующего ей элемента в векторе **V** вывести на экран символы **nr.**

Графическая иллюстрация этапов решения задачи для матрицы размером 6x6 приведена на следующем рисунке.

Исходная матрица M

6	8	6	1	8	4
5	9	0	5	8	7
3	3	8	6	0	5
1	8	**0**	**0**	8	9
3	**4**	**5**	**2**	**2**	7
4	**9**	**1**	**4**	**5**	**4**

Среднее = 3.33



Ступенчатый массив B

3	1	3
3		
0	0	1
1	0	2
0	2	
nr		

Максимальные элементы строк:

3	3	1	2	2	nr

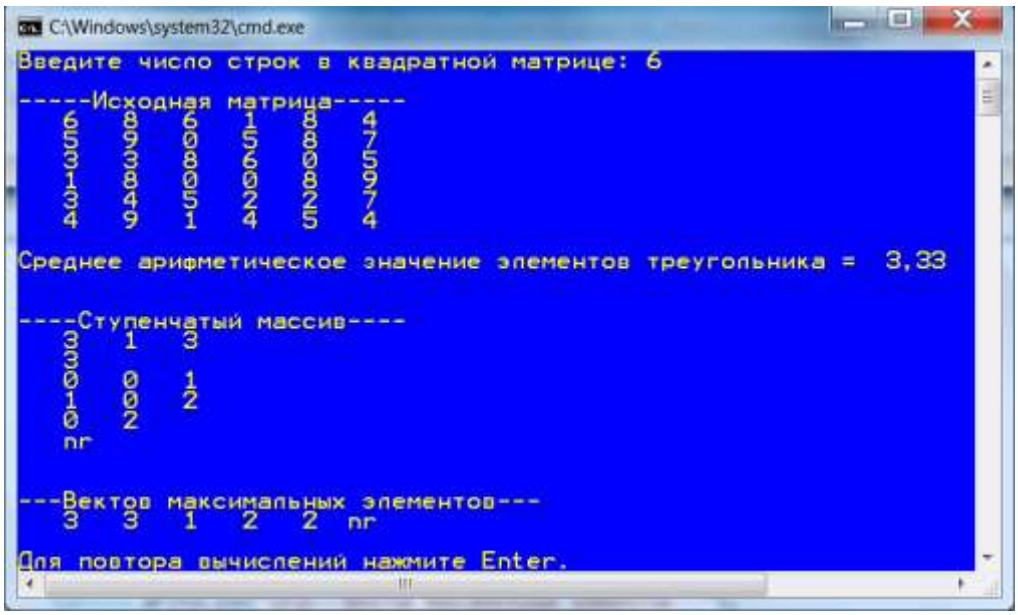
Примерное решение задачи 2 на языке C#:

```
using System;
```

```
class Program
{
   public static void Main()
   {
      Random rnd = new Random(17);         // Объект - генератор чисел

      do
      {
         int N;                  // Количество строк
         int[,] M;               // Ссылка на формируемую матрицу
         int[][] B;              // Ссылка на рваный массив
         int[] V;                        // Ссылка на вектор минимальных элементов.
         int[] K;                // Ссылка на массив с длинами строк.
         double sr;              // Среднее арифметическое значение элементов треугольника.
         int i, j, t;            // Номера строк и столбцов.

         Console.Write("Введите число строк в квадратной матрице: ");
         N = int.Parse(Console.ReadLine());

         M = new int[N, N];

         K = new int[N];
         V = new int[N];
         B = new int[N][];


         // Инициализация матрицы.
         for (i = 0; i < N; i++)
            for (j = 0; j < N; j++)
               M[i, j] = rnd.Next(0, 10);

         // Вывод матрицы.
         Console.WriteLine("\n-----Исходная матрица-----");
         for (i = 0; i < N; i++, Console.WriteLine())
            for (j = 0; j < N; j++)
               Console.Write("{0,4}", M[i, j]);

         //Вычисление среднего арифметического значения элементов треугольника.
         sr = 0;
         for (i = N / 2; i < N; i++)
            for (j = N - i - 1; j <= i; j++)
               sr += M[i, j];

         sr = sr / ((N * (N - 2) + N % 2) / 4 + N);         // Среднее арифметическое.

         Console.WriteLine(
               "\nСреднее арифметическое значение элементов треугольника = {0: 0.00}", sr);

         // Подсчитаем длину каждого столбца ступенчатого массива.
         for (j = 0; j < N; j++)
            for (i = 0; i < N; i++)
               if (M[i, j] < sr)
                  K[j]++;

         // Создадим строки ступенчатого массива.
         for (i = 0; i < N; i++)
            B[i] = new int[K[i]];

         // Скопируем элементы столбцов в строки ступенчатого массива.
         for (j = 0; j < N; j++)
            for (i = 0, t = 0; i < N; i++)
               if (M[i, j] < sr)
                  B[j][t++] = M[i, j];
```

```
        // Вывод массива.
        Console.WriteLine("\n\n----Ступенчатый массив----");

        for (i = 0; i < N; i++, Console.WriteLine())
        {
           if (B[i].Length == 0)
              Console.Write("   nr");

           for (j = 0; j < B[i].Length; j++)
              Console.Write("{0,4}", B[i][j]);
        }

        // Найдем макс. элементы.
        for (i = 0; i < N; i++)
        {
           if (B[i].Length > 0)
              V[i] = B[i][0];

           for (j = 0; j < B[i].Length; j++)
              if (V[i] < B[i][j])
                 V[i] = B[i][j];
        }

        // Вывод вектора максимальных элементов.
        Console.WriteLine("\n\n---Вектов максимальных элементов---");

        for (i = 0; i < N; i++)
           if (B[i].Length > 0)
              Console.Write("{0,4}", V[i]);
           else
              Console.Write("  nr");

        Console.WriteLine("\n\nДля повтора вычислений нажмите Enter.");
        Console.WriteLine("=====================================\n\n");
     } while (Console.ReadKey(false).Key == ConsoleKey.Enter);
  }
}
```

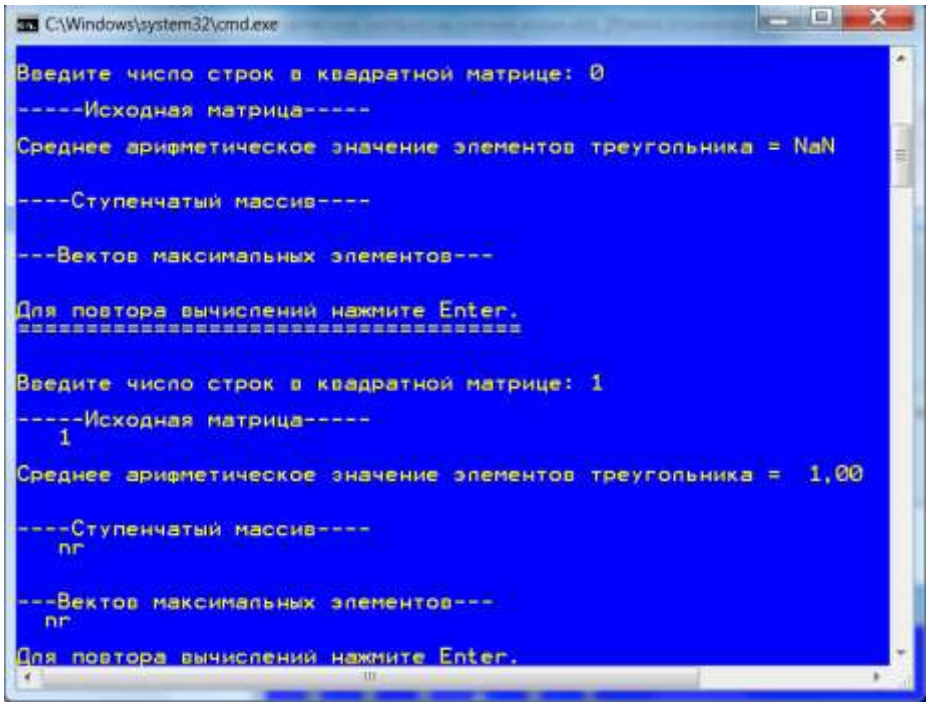
Вывод:



Примечание. Обратите внимание, как вычисляется количество элементов в треугольнике. Вывод массивов должен соответствовать табличной форме.

Так как массив **B** должен быть массивом массивов, то есть ступенчатым (рваным) массивом, то в этом массиве отдельные строки могут отсутствовать. Вместо отсутствующей строки ступенчатого массива и соответствующего ей значения в векторе V должно выводиться **nr** – нет строки (нет значения).

Программа должна работать как для четного значения N, так и для нечетного, как для N=0, так и для N=1. Для N=1 ступенчатый массив всегда пустой, поэтому всегда печатается **nr.** Например:



Задача 3. Вес 25 баллов

Рабочий получил задание от прораба купить в магазине N пар строительных перчаток. Отдельная пара перчаток стоит 30 руб., за связку из 10 пар перчаток дают скидку в 35 руб., а за упаковку из 10 связок – 270 руб. Составить программу, которая вычисляет, сколько необходимо купить упаковок, связок и отдельных пар перчаток, чтобы стоимость покупки была минимальной. Значение N ввести с клавиатуры.

В программе необходимо:

1. Найти оптимальный состав покупки без излишков, имеющей минимальную цену и состоящей ровно из N пар перчаток.

2. Попытаться выполнить оптимизацию покупки таким образом, чтобы стоимость покупки стала еще меньше, а общее количество покупаемых пар перчаток при этом стало больше введенного (>N). Стоимость должна рассматриваться как основной минимизируемый параметр.

3. Сравнить варианты 1 и 2 и вывести на экран полученную покупателем выгоду по стоимости покупки и общему количеству пар перчаток.

Особенность решения задачи состоит в том, что может быть сформирована оптимальная покупка без излишков и более дешевая покупка, но с излишками. Например, вместо 9 пар перчаток дешевле купить связку.

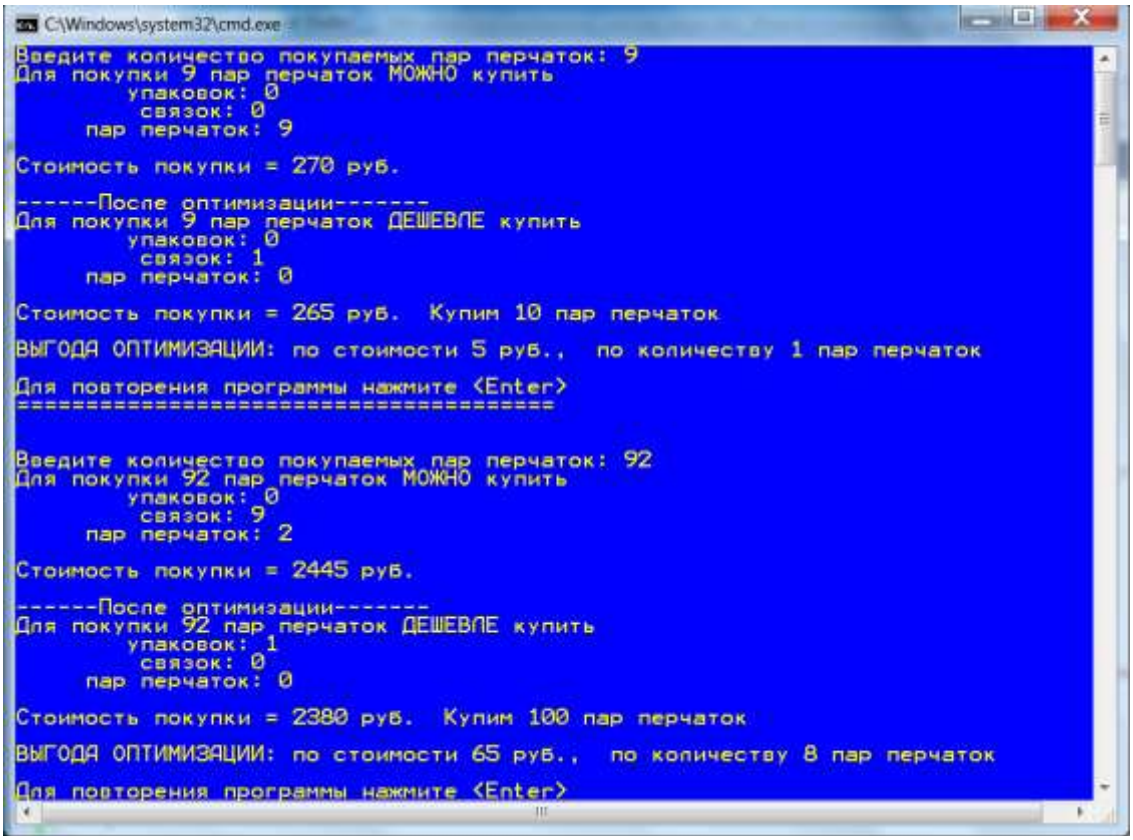
Примерное решение задачи 1 на языке C#:

```
using System;
class Program
```

```
{
   static void Main(string[] args)
   {
      int n1,     // Число упаковок.
          n2,     // Число связок.
          n3,     // Число пар.
          n,      // Покупаемое число пар.
          opt,    // Оптимальное количество пар перчаток.
          m,
          s1, s2;     // Стоимость покупки.
      do
      {

         while (true)
         {
            Console.Write("Введите количество покупаемых пар перчаток: ");
            if (int.TryParse(Console.ReadLine(), out n)) break;
            Console.WriteLine("\nВы ошиблись при вводе. Повторите!");
         }

         // Вычисляем оптимальную покупку для n пар перчаток.
         n1 = n / (100);     // Количество упаковок.
         m = n % 100;            // Остаток.
         n2 = m / 10;            // Количество связок.
         n3 = m % 10;            // Количество пар.

         Console.WriteLine("Для покупки {0} пар перчаток МОЖНО купить" +
                           "\n       упаковок: {1} \n       связок: {2} \n     пар перчаток: {3}",
                           n, n1, n2, n3);
         s1 = n3 * 30 + n2 * 265 + n1 * 2380;
         Console.WriteLine("\nСтоимость покупки = {0} руб.", s1);

         // Оптимизируем покупку для количества пар > n.
         if (n3 * 30 > 265)
         {
            n2++;       // Вместо 9 пар перчаток дешевле купить связку.
            n3 = 0;
         }

         if (n2 * 265 + n3 * 30 > 2380)
         {
            n1++;       // Вместо 9 пар и 10 связок дешевле купить упаковку.
            n2 = n3 = 0;
         }
         Console.WriteLine("\n------После оптимизации-------");

         Console.WriteLine("Для покупки {0} пар перчаток ДЕШЕВЛЕ купить" +
                           "\n       упаковок: {1} \n       связок: {2} \n     пар перчаток: {3}",
                           n, n1, n2, n3);

         s2 = n3 * 30 + n2 * 265 + n1 * 2380;
         opt = n1 * 100 + n2 * 10 + n3;
         Console.WriteLine("\nСтоимость покупки = {0} руб.  Купим {1} пар перчаток",
         s2, opt);

         Console.WriteLine("\nВЫГОДА ОПТИМИЗАЦИИ: по стоимости {0} руб.,  по количеству {1} пар перчаток", s1
- s2, opt - n);

         Console.WriteLine("\nДля повторения программы нажмите <Enter>");
         Console.WriteLine("=======================================\n\n");
      } while (Console.ReadKey(true).Key == ConsoleKey.Enter);

   }
}
```

Операторы, вычисляющие оптимальный состав покупки, выделены серым фоном.

Вывод:



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Всероссийская олимпиада школьников
«Миссия выполнима. Твое призвание-финансист!»

ОТБОРОЧНЫЙ (ЗАОЧНЫЙ) ЭТАП
Информатика 10-11 класс, 2015/2016 учебный год

Время выполнения заданий - 180 минут.

Задания с выборкой ответа

1. На столе находится 12 тарелок, из них 11 глубоких и одна мелкая. Укажите вариант формулы, которая позволяет определить количество информации в сообщении о том, что взятая наугад со стола тарелка является глубокой?

1) $\log_2 \frac{1}{12}$

2) $-\log_2 12$

3) $-\frac{11}{12}(\log_2 11 - \log_2 12)$

4) $-\left(\frac{11}{12}\log_2 11 - \log_2 12\right)$

5) $-\log_2 \frac{11}{12}$

2. Определены четыре множества: C={8,11,6,3,21}, D={2,11,3,8,14}, A={0, 3,11}, B={3,11,8,7}. Определить значение F = (C ∩ D) ∩ (A U B), где ∩ - знак операции пересечения, а U – знак операции объединения множеств.

1) {3, 11, 8}
2) {5,7}
3) {5,8}
4) Пустое множество

3. Укажите аналитическую запись логической функции, которой соответствует приведенная таблица истинности

1) $F = a \bullet \bar{b} \bullet \bar{c} + a \bullet \bar{b} \bullet c + a \bullet b \bullet c$
2) $F = a \bullet \bar{b} \bullet \bar{c} + a \bullet \bar{b} \bullet c$
3) $F = a \bullet \bar{b} \bullet \bar{c} + a \bullet b \bullet \bar{c} + a \bullet b \bullet c$
4) $F = a \bullet \bar{b} \bullet \bar{c} + a \bullet b \bullet \bar{c}$
5) $F = a \bullet \bar{b} \bullet \bar{c} + a \bullet b \bullet c$

Таблица истинности логической функции F=F(a,b,c)

a	**b**	**c**	**F**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Примечание: операция логического умножения обозначена знаком •, а операция логического сложения знаком +.

4. Результатом минимизации логической функции F=(x AND y AND z) OR (x AND y AND z) OR (x AND y AND z), где AND, OR, NOT – знаки логических операций конъюнкции, дизъюнкции и отрицания соответственно, является формула:

1) x AND y

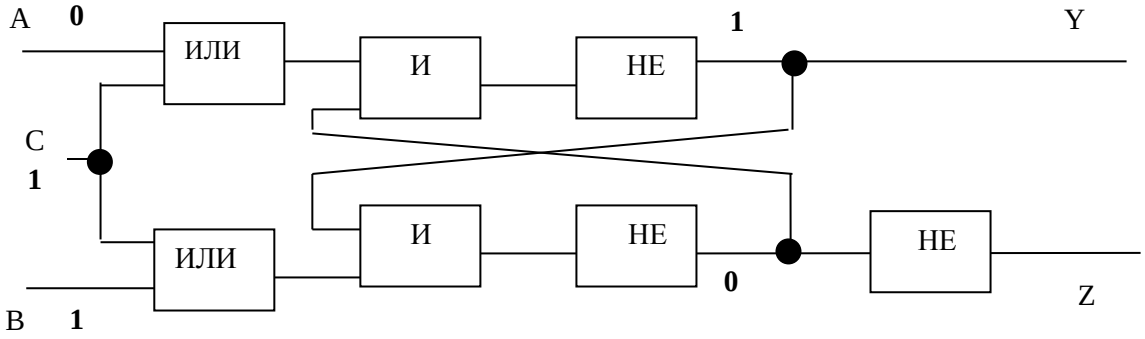
2) x AND y AND z
3) (x OR y) AND z AND x
4) y AND z OR x AND z
5) NOT x AND z OR X AND y

5. Для представления логической функции F(x,y,z) дана таблица истинности, в которой номера строк отсчитываются с нуля. Переменные в таблице истинности указываются слева направо в очередности: x,y,z. Первая строка имеет значения 0, 0, 0, вторая – 0, 0, 1 и т.д. Если логическая функция F(x,y,z) принимает значение "истина" в строке 1, то какая аналитическая форма представления (см. ниже) ей соответствует? (AND, OR и NOT – конъюнкции, дизъюнкции и отрицания соответственно.)

1) NOT x AND NOT y AND NOT z
2) (NOT x AND y AND NOT z) OR (NOT x AND y AND z) OR (x AND NOT y AND NOT z)
3) (NOT x AND NOT y AND NOT z) OR (NOT x AND y AND NOT z) OR (NOT x AND y AND z) OR (x AND NOT y AND NOT z)
4) (NOT x AND NOT y AND z) OR (NOT x AND y AND NOT z) OR (NOT x AND y AND z) OR (x AND NOT y AND NOT z)
5) NOT x AND NOT y AND z

6. Начальное состояние схемы с памятью приведено на рисунке. Какие логические сигналы необходимо установить на входах A,B,C, чтобы на выходах Y и Z были получены логические сигналы: Y=0 ,Z=0?

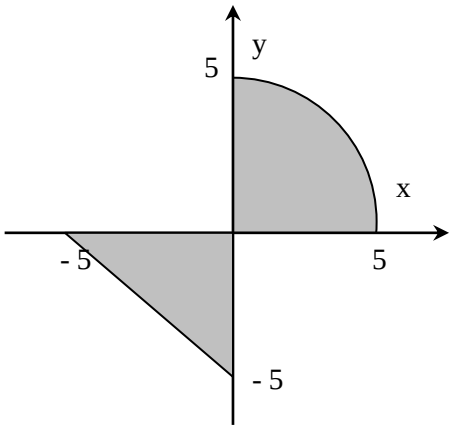
1) A=1, B=0, C=1
2) A=1, B=0, C=0
3) A=0, B=1, C=0



7. Определите выражение, которое соответствует геометрическому месту точек, представленному на рисунке в виде заштрихованной области. (and, or - знаки логических операций конъюнкции и дизъюнкции соответственно.)

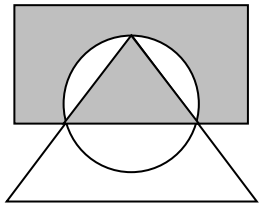
1) ((y <= x + 5) and (y >= - x – 5) and (x <= 0)) or ((25 >= x*x + y*y) and (x > 0) and (y >= 0))
2) ((25 >= x*x + y*y) and (x < 0) and (y >= 0)) or ((y >= x - 5) and (x >= 0) and (y < 0))
3) (25 >= x*x + y*y) and ((y >= x - 5) and (x >= 0) and (y < 0)

4) ((25 >= x*x + y*y) and (x >= 0) and (y >= 0)) or ((y >= - 5 - x) and (x < 0) and (y < 0))



8. Высказывания A, B, C истинны для точек, принадлежащих кругу, треугольнику и прямоугольнику соответственно. Определите выражение, которое соответствует геометрическому месту точек, представленному на рисунке в виде заштрихованной области. (and, or, not - знаки логических операций конъюнкции, дизъюнкции и отрицания соответственно.)

1) (B and A) or (C and A)
2) B and not (A and not C)
3) C and not (not B and A)
4) A and not (C and B)



9. В ячейке E2 таблицы MS Excel записана формула =$A$2+B$2+$C2+D2

	A	B	C	D	E
1					
2	2	4	6	3	15
3	7	3	2	2	

Формула из ячейки E2 была перемещена в ячейку E3, после чего была удалена строка 1.

	A	B	C	D	E
1	2	4	6	3	
2	7	3	2	2	?

Какая формула будет храниться в ячейке E2:

1) =$A$2+B$2+$C2+D1
2) =$A$1+B$1+$C2+D2
3) =$A$2+B$2+$C1+D1
4) =$A$1+B$1+$C1+D1
5) Будет выдано сообщение об ошибке

10. Алфавит состоит из 19 символов. Все символы кодируются одинаковым и минимально возможным количеством бит. Для хранения некоторого текста требуется 50 байтов памяти. Укажите, из скольких символов состоит текст.

1) 21
2) 80
3) 22
4) 100
5) 67

Задания с вычислением и записью ответа

11. Ниже приведены программы на четырех языках программирования, реализующие один и тот же алгоритм. Выберите одну из программ и определите, чему должна быть равна константа B, чтобы при выполнении программы было отпечатано 3 7.

Ответ: _____

PASCAL

```
program PR;
  const a = 9, B = ?;
  var  n, c, d : integer;
begin
  if a < B then
     n:=a;
  else
     n:=B;
  while (a mod n <> 0) or (B mod n <> 0) do
          n:=n-1;
  c := a div n;
  d := B div n;
  write(c,' ',d);
end.
```

C

```
#include "stdafx.h"

#define a  9
#define B  21

void main() {
   int n, c, d;
   for (n = (a < B) ? a : B; ; n--)
      if ((a % n == 0) & (B % n == 0)) break;
   c = a / n;
   d = B / n;
    printf("%d   %d ", c, d);
```

```
}
```

C#

```
class Program
{
  static void Main()
  {
     const int a = 9, B = ?;
     int n, c, d;
     for (n = (a < B) ? a : B; ; n--)
        if ((a % n == 0) & (B % n == 0)) break;
     c = a / n;
     d = B / n;
     System.Console.WriteLine(c + "  " + d);
  }
}
```

BASIC

```
A = 9: B = ?
IF A < B THEN
   N =A
ELSE
   N=B
ENDIF
WHILE A MOD N <> 0 OR B MOD N <> 0
   N=N-1
WEND
C = A \ N
D = B \ N
PRINT C & "  " & D
```

12. Какова длительность (в секундах) звучания моно аудио файла объёмом 240000 байт с частотой дискретизации 16 Кгц при 24-разрядном кодировании?
Ответ: _____

13. Используя запросы, приведенные в таблице, поисковый сервер выполнил поиск страниц. За время выполнения всех запросов состояние сети Интернет не менялось. В языке запросов символ «|» обозначает логическую операцию «ИЛИ», а «&» - логическую операцию «И». В таблице приведено количество найденных по запросам страниц.

Запрос	**Найдено страниц (в тысячах)**

Паскаль	180
Бэйсик	90
Питон	30
Паскаль & Бэйсик	20
Паскаль & Питон	10
Бэйсик & Питон	6
Паскаль & Бэйсик & Питон	2

Какое количество страниц (в тысячах) будет найдено по запросу
Паскаль | *Бэйсик* | *Питон*?
Ответ: _____

14. Используя всевозможные комбинации букв A, B, C, D, E, была составлена таблица пятибуквенных слов. В этой таблице все слова упорядочены по алфавиту и пронумерованы. Начало таблицы приведено ниже.

1. AAAAA
2. AAAAB
3. AAAAC
4. AAAAD
5. AAAAE
6. AAABA

……

Какой порядковый номер в указанной таблице будет иметь комбинация E,D,C,B,A?
Ответ: _____

15. Сколько единиц содержится в двоичной записи значения выражения:

$$2^{2050} + 4^{1000} - 2$$

Ответ: _____

16. Ниже на четырех языках программирования записана рекурсивная функция F. Какая последовательность чисел будет напечатана на экране при выполнении вызова F(3)? Все числа представить одной строкой без пробелов.

Бейсик

```
SUB F(n)
    IF n > 0 THEN
        F(n - 2)
        PRINT n
        F(n - 1)
    END IF
END SUB
```

Паскаль

```
procedure F(n: integer);
begin
        if n >0 then
        begin
                F(n - 2);
                writeln(n);
                F(n - 1);
        end;
end.
```

Си

```
void F(int n)  {
        if (n > 0) {
                F(n - 2);
                printf("%d\n", n);
                F(n - 1);
        }
}
```

C#

```
static void F(int n)
{
   if (n > 0)
   {
           F(n - 2);
           Console.WriteLine(n);
           F(n - 1);
   }
}
```

Ответ: _____

17. На автостоянке находится 1 автомобиль белого цвета, 3 серого цвета и 12 автомобилей черного цвета. Определите количество информации (в битах) в полученном сообщении, в котором указано, что выбранный наугад автомобиль оказался автомобилем не черного цвета.

Ответ: _____

18. Из четырех представленных ниже программ выберите одну, на знакомом Вам языке. Какие значения будут выведены при выполнении выбранной Вами программы. Если будет выведено несколько значений, то перечислить их через один пробел.

PASCAL

```
program PR;
const n = 5;
type mas=array[1..n,1..n] of integer;
var a:mas; i,j,s:integer;
begin
  for i:=1 to n do
    for j:=1 to n do
      a[i,j]: =(n + n) mod (i + j - 1)
 s:=0;
for i:= 2 to n do
   for j:= 1 to i-1 do
        s: = s + a[i, j];
        write(s)
end.
```

C

```
#include <stdio.h>
#define N  5
void main()  {
        int a[N][N], i, j, s=0;

        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
                a[i][j] = (N + N) % (i + j + 1);

        for (int i = 1; i < N; i++)
            for (int j = 0; j < i; j++)
                s = s + a[i][j];
                printf("%d", s);
}
```

C#

```
class Program
{
    static void Main()
    {
        const int N = 5;
        int[,] a = new int[N, N];
        int s = 0;

        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
```

```
            a[i, j] = (N + N) % (i + j + 1);

        for (int i = 1; i < N; i++)
            for (int j = 0; j < i; j++)
                s = s + a[i, j];
                System.Console.WriteLine(s);
    }
}
```

BASIC

```
n = 5
DIM a(n, n)
FOR i = 1 TO n
  FOR j = 1 TO n
     a(i, j) = (N + N) MOD (i + j - 1)
  NEXT j
NEXT i
s = 0
FOR i = 2 TO n
   FOR j= 1 TO i -1
       s = s + a(i, j)
   NEXT j
NEXT i
PRINT s
```

Ответ: _____

19. Восьмеричное число $A_8 = 534$ представлено в формате с фиксированной точкой в дополнительном коде. Длина формата – 12 двоичных разрядов. Определите, какому шестнадцатеричному числу будет соответствовать этот код после инвертирования содержимого всех битов и циклического сдвига влево на 4 двоичных разряда. Ответ запишите в виде шестнадцатеричного числа. Для записи шестнадцатеричных цифр со значением больше 9 используйте большие буквы латинского алфавита.
Ответ: _____

20. Определите максимально возможное количество пикселей в строке на экране монитора, если известно, что объем видеопамяти 480 Кбайт, количество строк – 1024, а код одного пикселя содержит 6 бит.
Ответ: _____