Требования к программам

1. Программа должна получать все параметры в качестве аргументов командной строки.
2. Задачи оцениваются независимо в двух группах: задачи 1–5 и задачи 6–10.
3. Аргументы командной строки **для задач 1–4**:
   1) n – размерность массива,
   2) p – количество выводимых значений в массиве,
   3) s – задает номер формулы для инициализации массива, должен быть равен 0 при вводе массива из файла,
   4) `filename` – имя файла, откуда надо прочитать массив. Этот аргумент **отсутствует**, если $s! = 0$.

   Например, запуск

   ```
   ./a01.out 4 4 0 a.txt
   ```

   означает, что массив длины 4 надо прочитать из файла `a.txt`, и выводить не более 4-х элементов массива, а запуск

   ```
   ./a01.out 1000000 6 1
   ```

   означает, что массив длины 1000000 надо инициализировать по формуле номер 1, и выводить не более 6-ти элементов массива.
4. Аргументы командной строки **для задач 5–8**: добавляется **дополнительный первый аргумент**, остальные аргументы – как в задачах 1–4:
   1) X – дополнительный аргумент: $X = k$ в задаче 5 (k – целое число), $X = x$ в задаче 6 (x – вещественное число), $X = b$ в задачах 7 и 8 (b – имя файла (текстовая строка)),
   2) n – размерность массива,
   3) p – количество выводимых значений в массиве,
   4) s – задает номер формулы для инициализации массива, должен быть равен 0 при вводе массива из файла,
   5) `filename` – имя файла, откуда надо прочитать массив. Этот аргумент **отсутствует**, если $s! = 0$.
5. Аргументы командной строки **для задач 9–10**: для задания двух входных массивов используется удвоенный "комплект" аргументов из задач 1–4:
   1) n – размерность массива a,
   2) p_a – количество выводимых значений в массиве a,
   3) s_a – задает номер формулы для инициализации массива a, должен быть равен 0 при вводе массива из файла,
   4) $\texttt{filename}_a$ – имя файла, откуда надо прочитать массив. Этот аргумент **отсутствует**, если $s_a! = 0$,

5) m – размерность массива b,

6) p_b – количество выводимых значений в массиве b,

7) s_b – задает номер формулы для инициализации массива b, должен быть равен 0 при вводе массива из файла,

8) `filename`$_b$ – имя файла, откуда надо прочитать массив. Этот аргумент **отсутствует**, если $s_b! = 0$.

Например, запуск

```
./a09.out 4 4 0 a.txt 6 6 0 b.txt
```

означает, что массив a длины 4 надо прочитать из файла `a.txt`, выводить не более 4-х элементов массива, массив b длины 6 надо прочитать из файла `b.txt`, выводить не более 6-ти элементов массива, запуск

```
./a09.out 1000000 6 1 8 6 0 b.txt
```

означает, что массив a длины 1000000 надо инициализировать по формуле номер 1, выводить не более 6-ти элементов массива, массив b длины 8 надо прочитать из файла `b.txt`, выводить не более 6-ти элементов массива, запуск

```
./a09.out 1000000 6 1 2000000 6 5
```

означает, что массив a длины 1000000 надо инициализировать по формуле номер 1, выводить не более 6-ти элементов массива, массив b длины 2000000 надо инициализировать по формуле номер 5, выводить не более 6-ти элементов массива.

6. Ввод массива должен быть оформлен в виде подпрограммы.

7. Ввод массива из файла. В указанном файле находится массив в формате:

число-1 число-2
...
число-n

где n - указанный размер массива. Числа разделяются либо пробелами, либо символом новой строки. Программа должна выводить сообщение об ошибке, если указанный файл не может быть прочитан, содержит меньшее количество данных или данные неверного формата.

8. Ввод массива по формуле. Элемент a_i массива A полагается равным

$$a_i = f(s,n,i), \quad i = 1,\ldots,n,$$

где $f(s,n,i)$ - функция, которая возвращает значение (i)-го элемента массива по формуле номер s (аргумент командной строки). Функция $f(s,n,i)$ должна быть оформлена в виде отдельной подпрограммы.

$$f(s,n,i) = \begin{cases} i & \text{при} \quad s = 1 \\ n-i & \text{при} \quad s = 2 \\ i/2.0 & \text{при} \quad s = 3 \\ n-i/2.0 & \text{при} \quad s = 4 \\ 2.0*i & \text{при} \quad s = 5 \\ n-2.0*i & \text{при} \quad s = 6 \end{cases}$$

9. Решение задачи должно быть оформлено в виде подпрограммы, получающей в качестве аргументов массив и его длину (в ряде задач также дополнительные аргументы). Получать в этой подпрограмме дополнительную информацию извне, а также выводить что-либо на экран, запрещается.

10. Программа должна содержать подпрограмму `print_array` вывода на экран массива длины не более p. Эта подпрограмма используется **для вывода исходного массива** после его инициализации, а также для вывода на экран результирующего массива, если массив изменялся. Подпрограмма выводит на экран не более, чем p элементов массива, где p – параметр этой подпрограммы (аргумент командной строки). Каждый элемент массива должен **печататься на новой строке**.

11. Вывод результата работы в функции `main` **для задач 1, 7, 8** должен производиться по формату:

```
printf ("%s : Task = %d Result = %d Elapsed = %.2f\n",
        argv[0], task, res, t);
```

где

- `argv[0]` – первый аргумент командной строки (имя образа программы),
- `task` – номер задачи,
- `res` – возвращаемое значение функции, реализующей решение этой задачи,
- `t` – время работы функции, реализующей решение этой задачи.

12. Вывод результата работы в функции `main` **для задач 3–5** должен производиться по формату:

```
printf ("New array:\n");
print_array (a, n, p); /* вывод нового состояния массива a */
printf ("%s : Task = %d Elapsed = %.2f\n",
        argv[0], task, t);
```

где

- `argv[0]` – первый аргумент командной строки (имя образа программы),
- `task` – номер задачи,
- `t` – время работы функции, реализующей решение этой задачи.

13. Вывод результата работы в функции `main` **для задач 6, 10** должен производиться по формату:

```
printf ("New array:\n");
print_array (a, res, p); /* вывод нового состояния массива a */
printf ("%s : Task = %d Result = %d Elapsed = %.2f\n",
        argv[0], task, res, t);
```

где

- `argv[0]` – первый аргумент командной строки (имя образа программы),
- `task` – номер задачи,

- `res` – возвращаемое значение функции, реализующей решение этой задачи,
- `t` – время работы функции, реализующей решение этой задачи.

14. Вывод результата работы в функции `main` **для задачи 9** должен производиться по формату:

```
printf ("New array:\n");
print_array (c, res, p); /* вывод массива с длины res */
printf ("%s : Task = %d Result = %d Elapsed = %.2f\n",
        argv[0], task, res, t);
```

где

- `argv[0]` – первый аргумент командной строки (имя образа программы),
- `task` – номер задачи,
- `res` – возвращаемое значение функции, реализующей решение этой задачи,
- `t` – время работы функции, реализующей решение этой задачи.

Задачи

1. Написать функцию, получающую в качестве аргументов массив вещественных чисел и целое число, являющееся длиной этого массива, и возвращающую целое значение, равное 1, если массив симметричный, и 0 в противном случае. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту функцию и выводить на экран результат ее работы.

2. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел и целое число, являющееся длиной этого массива, и переставляющую элементы массива в обратном порядке, произведя при этом не более $3n/2 + O(1)$ перемещений элементов. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

3. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел и целое число, являющееся длиной этого массива, и заменяющую каждый элемент массива (кроме первого и последнего) на полусумму соседей. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

4. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел и целое число n, являющееся длиной этого массива, и циклически сдвигающую элементы массива на одну позицию вправо, произведя не более $n + O(1)$ перемещений элементов. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

5. Написать подпрограмму, получающую в качестве аргументов целое число k, массив вещественных чисел, целое число n, являющееся длиной этого массива, и циклически сдвигающую элементы массива на k позиций вправо произведя не более $n + O(1)$ перемещений элементов. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

6. Написать функцию, получающую в качестве аргументов вещественное число x, массив $a[n]$ вещественных чисел, целое число n, являющееся длиной этого массива, и выбрасывающую из массива все элементы, меньшие x (при выбрасывании элемента $a[i]$ все элементы с номером, большим i, сдвигаются на одну позицию к началу массива и длина массива уменьшается на 1), произведя при этом не более $n+O(1)$ перемещений элементов. Функция возвращает длину получившегося в результате массива. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту функцию и выводить на экран результат ее работы.

7. Написать функцию, получающую в качестве аргументов имя файла b, массив $a[n]$ различных вещественных чисел, целое число n, являющееся длиной этого массива, и возвращающую целое число, равное количеству элементов в последовательности из файла b, совпадающих с каким-либо элементом из массива $a[n]$. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл b, прочитать элемент и т.д.. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту функцию и выводить на экран результат ее работы.

8. Написать функцию, получающую в качестве аргументов имя файла b, массив $a[n]$ различных вещественных чисел, целое число n, являющееся длиной этого массива, и возвращающую целое число, равное количеству вхождений подпоследовательности $a[n]$ в последовательность, содержащуюся в заданном файле b. Под вхождением понимается совпадение n подряд идущих элементов последовательности в файле b с элементами массива $a[0],\dots a[n-1]$. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл b, прочитать элемент и т.д.. Основная программа должна заполнять данными массив, выводить его на экран, вызывать эту функцию и выводить на экран результат ее работы.

9. Написать функцию, получающую в качестве аргументов три массива $a[n]$, $b[m]$, $c[n+m]$ вещественных чисел, где $a[n]$, $b[m]$ строго возрастают, и целые числа n и m, и строящую по строго возрастающим массивам $a[n]$, $b[m]$ строго возрастающий массив $c[n+m]$ слиянием первых двух (с выбрасыванием повторяющихся чисел) за не более чем $2(n+m)+O(1)$ сравнений и $n+m+O(1)$ пересылок элементов по следующему алгоритму

   Просматриваем очередные элементы $a[i]$, $i=0,\dots,n-1$ и $b[j]$, $j=0,\dots,m-1$ массивов a и b. Если $a[i]<b[j]$, то $c[k]=a[i]$ и увеличиваем i на 1, если $a[i]>b[j]$, то $c[k]=b[j]$ и увеличиваем j на 1, иначе увеличиваем на 1 i или j. Здесь k, $0\le k\le n+m-1$ – очередной элемент массива c.

   Функция должна возвращать длину получившегося в результате массива c. Основная программа должна заполнять данными массивы a и b, выводить их на экран, вызывать эту подпрограмму и выводить на экран результат ее работы – массив c.

10. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ длины n, массив $b[m]$ длины m и целые числа n и m, и выбрасывающую все элементы массива a, равные какому-либо элементу массива b (при выбрасывании элемента $a[i]$ все элементы с номером, большим i, сдвигаются на одну позицию к началу массива и длина массива уменьшается на 1), произведя при этом не более $n+O(1)$ перемещений элементов. Функция возвращает длину получившегося в результате массива. Основная программа должна заполнять данными массивы a и b, выводить их на экран, вызывать эту подпрограмму и выводить на экран результат ее работы – новое состояние массива a.