

ПРАВИЛА

1. Для задач ниже создать один проект в IntelliJ Idea. Папку этого проекта целиком поместить в репозиторий. Также поместить туда в корень файл .gitignore (выложу его в открытый репозиторий). Далее добавить все файлы (можно с помощью `git add .` (с точкой) - это добавит все папку), закоммитить, запустить
2. (касается задач, в которых присутствует класс с методом `main`) — для каждого такого задания создается отдельный файл `TaskNNN.java`, где `NNN` - трехзначный номер задачи. Файл `TaskNNN` содержит метод `main`. Для вспомогательных классов, используемых в задаче, можно и даже желательно создавать отдельные `java`-файлы.
3. `.java` код надо подписывать в самом верху следующим образом (привожу пример по себе на примере своей группы 953 и задачи 000):

```
/**
 * @author Mikhail Abramskiy
 * 953a
 * 000 (для вспомогательного класса указывайте для чего используется,
 *      например for 001, 002 and 007)
 */
```

- 001 Построить автомат, который проверяет, что у двоичной последовательности последние два символа - единицы. Написать код `java` для этого автомата (без оптимизации).
- 002 Построить автомат, который проверяет, что входное слово имеет вид: `00...011...100...0`, т.е. состоит из трех частей - сначала нули, потом единицы, потом снова нули. Некоторые части могут отсутствовать - такое слово тоже надо принимать.
- 003 Построить автомат, решающий предыдущую задачу, но теперь все три части должны в слове присутствовать, а иначе слово не принимается.
- 004 Построить автомат, который проверяет, что двоичная последовательность представляет собой четное число. Написать код `java` для этого автомата (без оптимизации).
- 005 Построить автомат, который проверяет, что входное слово имеет четное число и нулей и единиц.
- 006 Построить автомат, проверяющий, что слово начинается на два разных символа, а заканчивается на два одинаковых. Длина слова должна быть ≥ 4 .

Для следующих регулярных выражений нарисуйте диаграмму автомата, распознающего язык, представленный регулярным выражением.

Обратите внимание — иногда в регулярных выражениях (в их теоретических аспектах) символ `+` используют в качестве операции `|` (или). Обращайте внимание на то, стоит ли плюс на одном уровне с символами или находится немного в степени. Пример: `a + b` это "а или b". `a+b` это "несколько а подряд, потом один b".

Также напоминаю, что `ε` обозначает пустое слово (пустую строку)

007 $(11 + 0)^*(00 + 1)^*$

008 $(1 + 01 + 001)^*(\epsilon + 0 + 00)$

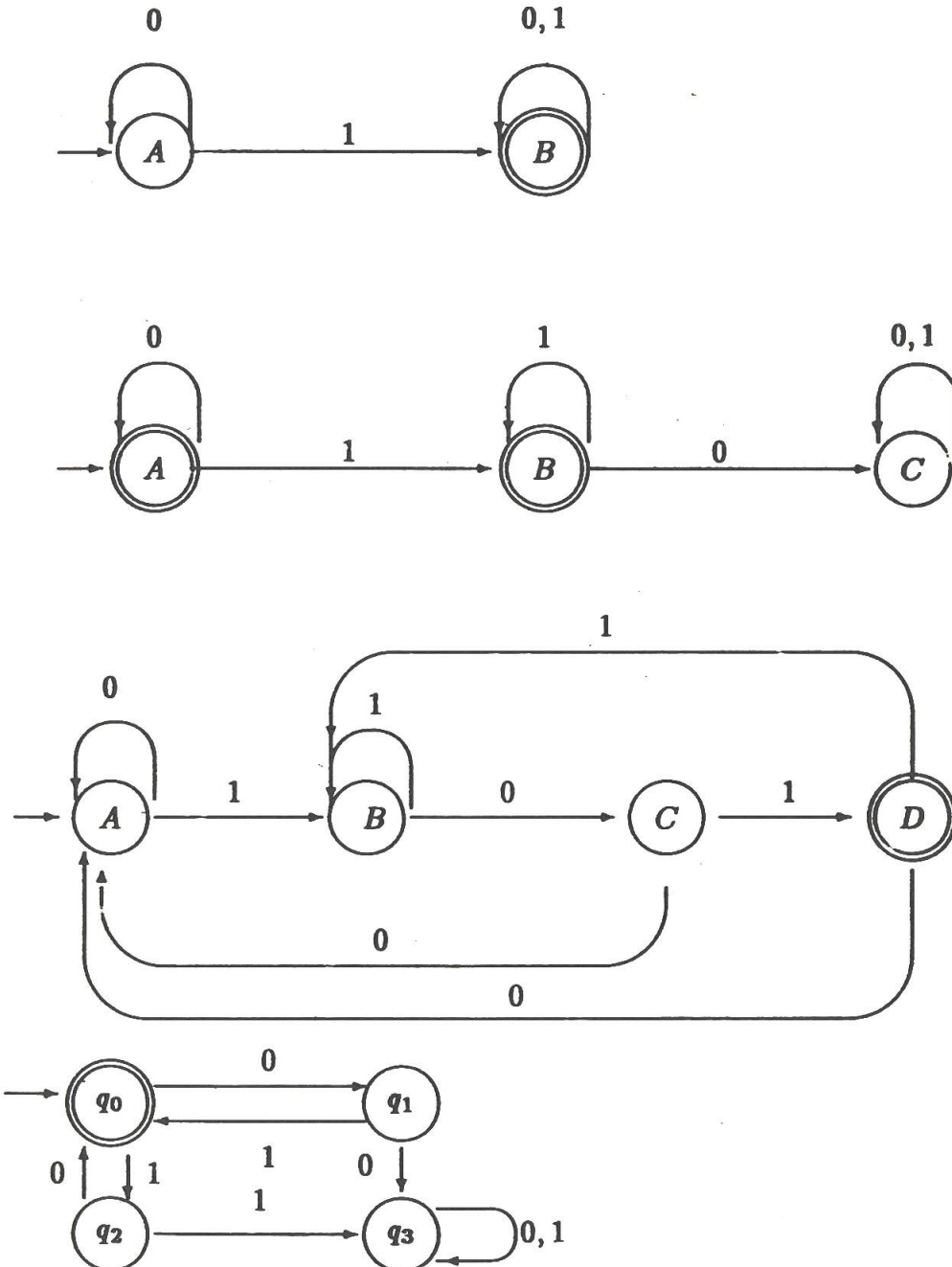
009 $(00 + 11 + (01 + 10)(01 + 11)^*(01 + 10))^*$

010 $10 + (0 + 11)0^*1$

011 $01(((10)^* + 111)^* + 0)^*1$

012-015 Для диаграмм на рисунке написать регулярное выражение, соответствующее языку, который каждый автомат распознает.

Затем построить таблицу перехода и реализовать указанные автоматы в программе на языке java. Разумеется, вы можете перенумеровать состояния, чтобы удобно было реализовывать.



Написать регулярное выражение, реализовать его в java-программе (использовать Pattern и Matcher), код залить в свой репозиторий

016 Десятичное число, которое содержит более 3 и менее 6 четных цифр и ни одной нечетной.

017 Десятичное число, в котором нет трех четных цифр подряд.

018 Десятичное число, в котором нет ни двух четных, ни двух нечетных цифр подряд (например, 12345678).