Лабораторная работа №7.

1. Решения каждого задания должны находиться в папке лиги в файле с соответствующем названием (Task1 – Task5). Создавать все требуемые по заданию классы и структуры необходимо внутри соответствующего класса (Task1 – Task5). Автотестами будет сравниваться состояние Ваших объектов с выходными данными.
2. Названия всех свойств и сигнатуры методов должны **в точности совпадать** с теми, что указаны в задании, так как они используются в тестах. Все поля должны быть приватными. Все поля необходимо инициализировать в конструкторе(-ах). В свойствах и методах перед обращением к ссылочным полям проверять их на *null*. Вам нужно решить задания в точности так, как указано в условии. Для выполнения задания вы можете использовать дополнительные **приватные** поля, свойства или методы.
3. Для проверки правильности решения задания необходимо найти соответствующий файл с тестами в папке Lab7Test, раскомментировать его содержимое и запустить автоматическое тестирование. Самостоятельное тестирование номеров перед сдачей работы на GitHub является обязательным для получения допуска. Также для ручной проверки и поиска своих ошибок, которые были выявлены при автоматическом тестировании, можно использовать файл Program.cs. При тестировании содержимое Program.cs не проверяется.
4. Если все тесты пройдены успешно, Вы можете отправить лабораторную на заключительную проверку на GitHub. Обновите все измененные файлы в своем репозитории (свои **решения** и **раскомментированные тесты**). После этого отправьте Pull Request в вашу ветку на финальную проверку. Более подробная инструкция описана в задании на Moodle.
5. Если Ваша работа принята, можно прорешать номера из других лиг в качестве подготовки к контрольной. Рекомендуется прорешать больше заданий тем, кто не имеет опыта в программировании и был определен в белую лигу, потому что для успешного освоения курса Вам нужно научиться решать задачи уровня зеленой и частично синей лиги.
6. В работе разрешены методы следующих классов: *System, Console, Math, Random, Convert, Array, Linq*.

Задания белой лиги.

Task1. Результаты соревнований по прыжкам в длину определяются по сумме двух попыток.

Создать публичную структуру **Participant** с приватными полями для фамилии, клуба и результатов двух прыжков. Создать публичные свойства только для чтения каждого из приватных полей: **Surname, Club, FirstJump, SecondJump**. Создать публичное свойство **JumpSum**, которое возвращает результат суммы двух прыжков.

Конструктор должен принимать 2 строковых поля: фамилию и название клуба. Создать метод **public void Jump(double result)**, который заполняет результат первого и второго прыжка данными. Если они уже заполнены, то новыми данными их перезаписывать не нужно. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию суммы обеих попыток. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task2. Результаты соревнований по прыжкам в высоту определяются по лучшей из двух попыток.

Создать публичную структуру **Participant** с приватными полями для имени, фамилии и результатов двух прыжков. Создать публичные свойства только для чтения каждого из приватных полей: **Name, Surname, FirstJump, SecondJump**. Создать публичное свойство **BestJump**, которое возвращает результат лучшего прыжка спортсмена.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Создать метод **public void Jump(double result)**, который заполняет результат первого и второго прыжка данными. Если они уже заполнены, то новыми данными их перезаписывать не нужно. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию лучшего результата спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task3. Группе студентов в результате полусеместровой аттестации были выставлены оценки по информатике, а также определено количество пропущенных занятий. Успеваемость каждого студента оценивается следующими баллами: «0» (не аттестован), «2», «3», «4» или «5».

Создать публичную структуру **Student** с приватными полями для имени, фамилии, массива оценок и количества пропусков. Создать публичные свойства только для чтения приватных полей **Name, Surname, Skipped**. Создать публичное свойство **AverageMark**, возвращающее среднюю оценку по предметам.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Создать метод **public void Lesson(int mark)**, который вставляет в массив оценок новую оценку, если она отлична от 0. Вместо добавления оценки 0 нужно увеличить количество пропусков на 1. Создать статический метод для сортировки массива структур **public static void SortBySkipped(Student[] array)** в порядке убывания количества пропущенных ими занятий. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task4. После окончания соревнования по шахматам турнирная таблица содержит фамилии участников и результаты сыгранных партий (выигрыш – 1 очко, ничья – 1/2 очка, проигрыш – 0 очков). Составить итоговую таблицу в порядке убывания полученных участниками очков.

Создать публичную структуру **Participant** с приватными поля для имени, фамилии, массива очков, полученных за матчи. Создать публичные свойства только для чтения приватных полей: **Name**, **Surname**, **Scores**. Создать публичное свойство только для чтения **TotalScore**, которое выводит сумму всех очков спортсмена.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Также в нем проинициализировать массив очков. Создать метод **public void PlayMatch(double result)**, который добавляет в массив игр результат очередного матча. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию общего результата спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task5. Обработать результаты первенства по футболу. Результаты каждой игры заданы в виде названий команд и счета (количество забитых и пропущенных мячей). Сформировать таблицу очков (выигрыш – 3, ничья – 1, проигрыш – 0) и упорядочить результаты в соответствии с занятым местом. Если сумма очков у двух команд одинакова, то сравниваются разности забитых и пропущенных мячей.

Создать публичную структуру **Match** с приватными полями забитых и пропущенных мячей. Создать публичные свойства только для чтения приватных полей: **Goals, Misses** и свойства **Difference** и **Score**, которые возвращают разность забитых и пропущенных голов и количество очков за матч соответственно.

Конструктор должен принимать 2 строковых поля: забитые и пропущенные голы. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Создать публичную структуру **Team** с приватными полями названия и массива матчей. Создать публичные свойства только для чтения приватных полей **Name, Matches**, свойства **TotalDifference** и **TotalScore**, которые возвращают суммарную разность забитых и пропущенных голов во всех матчах и суммарное количество баллов, набранных командой, соответветственно.

Конструктор должен принимать строковое поле названия команды и инициализировать массив матчей. Создать метод **public void PlayMatch(int goals, int misses)**, который добавляет новый матч в массив матчей команды. Создать метод для сортировки массива команд **public static void SortTeams(Team[] teams)** в порядке убывания суммарных очков команд. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Задания зеленой лиги.

Task1. Составить программу для обработки результатов кросса на 500 м для женщин.

Создать публичную структуру **Participant** с приватными полями для фамилии, группы, фамилия тренера и результата забега. Создать статическое неизменяемое поле для норматива и статическое поле для счетчика людей, прошедших норматив.

Создать публичные свойства только для чтения каждого из приватных полей: **Surname, Group, Trainer, Result**. Создать статическое публичное свойство **PassedTheStandard**, возвращающее количество прошедших норматив участниц, хранимое в поле **_passed**. Создать публичное логическое свойство только для чтения **HasPassed**, которое возвращает прошла участница норматив или нет.

Статический конструктор должен установить значение норматива 100, а значение счетчика прошедших норматив участниц 0. Конструктор должен принимать 3 строковых поля: фамилию, группу и фамилию тренера.

Создать метод **public void Run(double result)**, который заполняет результат участницы и увеличивает количество прошедших норматив, если он был выполнен. Если результат уже заполнен, то новыми данными его перезаписывать не нужно. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task2. Студенты одной группы в сессию сдают четыре экзамена. Составить список студентов, средний балл которых по всем экзаменам не менее «4».

Создать публичную структуру **Student** с приватными полями имени, фамилии и массива оценок по 4-м предметам (оценки варьируются от 2 до 5). Создать публичные свойства только для чтения приватных полей **Name, Surname, Marks,** свойство **AverageMark**, которое возвращает среднее значение оценок студента, и свойство **IsExcellent**, которое возвращает значение, все ли оценки по предметам «4» и выше.

Конструктор должен принимать 2 строковых поля (имя и фамилию) и инициализировать массив оценок по предметам нулями. Создать метод **public void Exam(int mark)**, который заменяет оценку по предмету новой оценкой. Каждый предмет студент может сдать только 1 раз. Создать статический метод для сортировки массива структур **public static void SortByAverageMark(Student[] array)** в порядке убывания среднего балла. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task3. Группа учащихся подготовительного отделения сдает выпускные экзамены по трем предметам (математика, физика, русский язык). Учащийся, получивший "2", сразу отчисляется.

Создать публичную структуру **Student** с приватными полями для имени, фамилии, массива оценок по 3-м предметам (оценки варьируются от 2 до 5) и статуса его обучения (отчислен ли студент). Создать публичные свойства только для чтения приватных полей **Name, Surname, Marks**, **IsExpelled** и свойство **AverageMark**, которое возвращает среднее значение оценок студента.

Конструктор должен принимать 2 строковых поля: имя и фамилию и инициализировать массив оценок по предметам нулями. Создать метод **public void Exam(int mark)**, который заменяет оценку по предмету новой оценкой, если эта оценка выше «2». В противном случае студента нужно отчислить и его оценки больше не должны меняться. Создать статический метод для сортировки массива структур **public static void SortByAverageMark(Student [] array)** в порядке убывания среднего балла. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task4. На основании результатов соревнований по прыжкам в длину составить итоговый протокол соревнований, считая, что в зачет идет лучший результат.

Создать публичную структуру **Participant** с приватными полями для имени, фамилии и массива из трех попыток. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Jumps**. Создать публичное свойство **BestJump**, которое возвращает результат лучшего прыжка спортсмена.

Конструктор должен принимать 2 строковых поля: имя и фамилию и инициализировать массив прыжков. Создать метод **public void Jump(double result)**, который заполняет результат очередного прыжка в массиве данными. Если они уже заполнены, то новыми данными их перезаписывать не нужно. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию лучшего результата спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task5. Результаты сессии содержат оценки 5 экзаменов по каждой группе. Определить средний балл для трех групп студентов одного потока и выдать список групп в порядке убывания среднего балла.

Создать публичную структуру **Student** с приватными полями имени, фамилии и массива оценок по 5-м предметам (оценки варьируются от 2 до 5). Создать публичные свойства только для чтения приватных полей **Name, Surname, Marks** и свойство **AverageMark**, которое возвращает среднее значение оценок студента.

Конструктор должен принимать 2 строковых поля: имя и фамилию и инициализировать массив оценок по предметам. Создать метод **public void Exam(int mark)**, который заменяет оценку по предмету новой оценкой. Каждый предмет студент может сдать только 1 раз. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Создать публичную структуру **Group** с приватными полями названия и массива студентов. Создать публичные свойства только для чтения приватных полей **Name, Students** и свойство **AverageMark**, которое возвращает среднее значение оценок всех студентов группы.

Конструктор должен принимать строковое поле названия группы и инициализировать массив студентов. Создать два метода **public void Add**, которые позволяют добавить одного и несколько студентов в группу соответственно. Создать статический метод для сортировки массива структур **public static void SortByAverageMark(Group[] array)** в порядке убывания среднего балла в группе. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Задания синей лиги.

Task1. Радиокомпания провела опрос слушателей по вопросу: «Кого вы считаете человеком года?». Определить частоту встречающихся ответов за каждого кандидата.

Создать публичную структуру **Response** с приватными полями для хранения имени и фамилии, и счетчика количества голосов за данного человека. Создать публичные свойства только для чтения каждого из приватных полей: **Name, Surname, Votes**.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Создать публичный метод **public int CountVotes(Response[] responses)**, который будет подсчитывать, сколько голосов из общего списка ответов соответствуют текущему кандидату (он(а) также находится в списке). Метод должен обновлять счетчик количества голосов за данного человека у всех объектов, соответствующих данному кандидату. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task2. Протокол соревнований по прыжкам в воду содержит список фамилий спортсменов и баллы, выставленные 5 судьями по результатам 2 прыжков. Получить итоговый протокол, в порядке занятых спортсменами мест по результатам обоих прыжков.

Создать публичную структуру **Participant** с приватными поля для имени, фамилии, матрицы из пяти оценок судьей по обоим прыжкам. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Marks**. Создать публичное свойство только для чтения **TotalScore**, которое выводит сумму всех оценок спортсмена.

Конструктор должен принимать 2 строковых поля: имя и фамилию и инициализировать оценки судей нулями. Создать метод **public void Jump(int[] result)**, который заполняет результат очередного прыжка оценками судей. Если они уже заполнены, то новыми данными их перезаписывать не нужно. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию суммарного результата спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task3. Для формирования сборной по хоккею предварительно отобрано 30 игроков. На основании протоколов игр составлена таблица, в которой содержится штрафное время каждого игрока по каждой игре (0, 2, 5 или 10 мин). Написать программу, которая составляет список кандидатов в сборную в порядке возрастания суммарного штрафного времени. Игрок, оштрафованный на 10 мин хотя бы в одном матче, из списка кандидатов исключается.

Создать публичную структуру **Participant** с приватными поля для имени, фамилии, массива штрафных минут, полученных за матчи. Создать публичные свойства только для чтения приватных полей: **Name, Surname, PenaltyTimes**. Создать публичное свойство только для чтения **TotalTime**, которое выводит сумму всех штрафных минут спортсмена. Создать публичное свойство только для чтения **IsExpelled**, которое показывает, исключен ли спортсмен из списков кандидатов в сборную.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Также в нем проинициализировать массив минут. Создать метод **public void PlayMatch(int time)**, который добавляет в массив штрафов штрафное время в очередном матче. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по возрастанию общего штрафного времени спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task4. Соревнования по футболу между командами проводятся в два этапа. Для проведения первого этапа участники разбиваются на две группы по 12 команд. Для проведения второго этапа

выбирается 6 лучших команд каждой группы по результатам первого этапа. Составить список команд участников второго этапа.

Создать публичную структуру **Team** с приватными полями названия команды, массива очков, полученных за матчи. Создать публичные свойства только для чтения приватных полей: **Name, Scores**. Создать публичное свойство только для чтения **TotalScore**, которое выводит сумму набранных командой очков.

Конструктор должен принимать строковое поле: название команды. Также в нем проинициализировать массив очков. Создать метод **public void PlayMatch(int result),** который добавляет в массив игр результат очередного матча. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Создать публичную структуру **Group** с приватным полем названия и массива спортсменов. Создать публичные свойства только для чтения приватных полей **Name** и **Teams.**

Конструктор должен принимать строковое поле названия группы и инициализировать массив команд. Создать два метода **public void Add**, которые позволяют добавить одного и несколько команд в группу соответственно. Но в группе должно оказаться не более 12 команд. Создать метод для сортировки массива команд **public void Sort()** в порядке убывания суммарных очков. Создать статический метод для слияния двух отсортированных массивов команд двух групп в новую группу с ограничением по размеру **public static Group Merge(Group group1, Group group2, int size)**. Название соединенной группы должно быть “Финалисты”. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task5. В соревнованиях участвуют три команды по 6 человек. Результаты соревнований представлены в виде мест участников каждой команды (1 - 18). Определить команду – победителя, вычислив количество баллов, набранное каждой командой. Участнику, занявшему 1-е место, начисляется 5 баллов, 2-е – 4, 3-е – 3, 4-е – 2, 5-е – 1, остальным – 0 баллов. При равенстве баллов победителем считается команда, за которую выступает участник, занявший 1-е место.

Создать публичную структуру **Sportsman** с приватными полями имени, фамилии, занятого места. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Place**.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Создать метод **public void SetPlace(int place),** который устанавливает место, которое занял спортсмен в таблице. Место должно устанавливаться только 1 раз. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Создать публичную структуру **Team** с приватными полями названия и массива спортсменов. Создать публичные свойства только для чтения приватных полей **Name, Sportsmen**, свойство **TotalScore**, которое возвращает количество баллов, набранных командой в зависимости от мест, занятых её участниками, и свойство **TopPlace**, которое выводит наивысшее место, которое занял кто-то из членов команды.

Конструктор должен принимать строковое поле названия команды и инициализировать массив спортсменов. Создать два метода **public void Add**, которые позволяют добавить одного и несколько спортсменов в группу соответственно. Создать метод для сортировки массива команд **public static void Sort(Team[] teams)** в порядке занятых мест (принцип описан в начале номера). Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Задания фиолетовой лиги.

Task1. В соревнованиях по прыжкам в воду оценивают 7 судей. Каждый спортсмен выполняет 4 прыжка. Каждый прыжок имеет одну из четырех категорий сложности, оцениваемую коэффициентом (от 2,5 до 3,5). Качество прыжка оценивается судьями по 6-балльной целочисленной шкале. Далее лучшая и худшая оценки отбрасываются, остальные складываются, и сумма умножается на коэффициент сложности. Получить итоговую таблицу в порядке занятых мест.

Создать публичную структуру **Participant** с приватными поля для имени, фамилии, массива из четырех коэффициентов и матрицы оценок судей четырех прыжков. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Coefs** и **Marks**. Создать публичное свойство только для чтения **TotalScore**, которое выводит результат по итогам 4-х прыжков.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Также в нем проинициализировать коэффициенты значениями 2.5 и оценки прыжков нулями. Создать метод **public void SetCriterias(double[] coefs)**, который заполняет четыре коэффициента. Создать метод **public void Jump(int[] marks)**, который заполняет результат очередного прыжка в матрице оценками судей. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию суммарного результата спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task2. Соревнования по прыжкам на лыжах со 120-метрового трамплина судят 5 судей. Каждый судья выставляет оценку за стиль прыжка по 20-балльной шкале. Меньшая и большая оценки отбрасываются, остальные суммируются. К этой сумме прибавляются очки за дальность прыжка: 120 метров – 60 очков, за каждый метр превышения добавляются по 2 очка, при меньшей дальности отнимаются 2 очка за каждый метр (очков не может стать меньше нуля). Получить итоговую таблицу соревнований, содержащую фамилию и итоговый результат для каждого участника в порядке занятых мест.

Создать публичную структуру **Participant** с приватными поля для имени, фамилии, расстояния прыжка, массива из пяти оценок судьей. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Distance** и **Marks**. Создать публичное свойство только для чтения **Result**, которое выводит итоговый результат прыжка.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Также в нем проинициализировать оценки судей нулями. Создать метод **public void Jump(int distance, int[] marks)**, который заполняет результат прыжка и оценки судей. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по убыванию суммарного результата спортсмена. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task3. Результаты соревнований фигуристов по одному из видов многоборья представлены оценками семи судей в баллах (от 0,0 до 6,0). По результатам оценок судьи определяется место каждого участника у этого судьи. Места участников определяются далее по сумме мест, которые каждый участник занял у всех судей. Составить программу, определяющую по исходной таблице оценок фамилии и сумму мест участников в порядке занятых ими мест.

Создать публичную структуру **Participant** с приватными поля для имени, фамилии, массива оценок за прыжки, массива мест, полученных спортсменом у судей, наивысшего места, полученного у судей и суммы полученных мест. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Marks, Places, TopPlace** и **TotalMark**. Создать публичное свойство только для чтения **Score**, которое выводит итоговое место спортсмена как сумму занятых им мест.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Также в нем проинициализировать массивы оценок судей и занятых мест нулями. Создать метод **public void Evaluate(double result)**, который добавляет оценку очередного судьи в массив. Если они уже заполнены, то новыми данными их перезаписывать не нужно. Создать метод **public static void SetPlaces(Participant[] participants)**, который заполняет массив мест у спортсменов, занятых у каждого судьи. На выходе из метода массив должен быть отсортирован по возрастанию мест у последнего судьи. Создать статический метод для сортировки массива структур **public static void Sort(Participant[] array)** по возрастанию суммы занятых мест. При равенстве выше должен оказаться спортсмен, который занимал более высокое место хотя бы у одного судьи. При равенстве и по этому критерию, выше должен быть спортсмен, у которого больше суммарный результат очков, выставленных судьями. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task4. Лыжные гонки проводятся отдельно для двух групп участников. Результаты соревнований заданы в виде фамилий участников и их результатов в каждой группе. Расположить результаты соревнований в каждой группе в порядке занятых мест. Объединить результаты обеих групп с сохранением упорядоченности.

Создать публичную структуру **Sportsman** с приватными полями имени, фамилии, времени бега. Создать публичные свойства только для чтения приватных полей: **Name, Surname, Time**.

Конструктор должен принимать 2 строковых поля: имя и фамилию. Создать метод **public void Run(double time)**, который устанавливает время, за которое спортсмен пробежал гонку. Время должно устанавливаться только 1 раз. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Создать публичную структуру Group с приватным полем названия и массива спортсменов. Создать публичные свойства только для чтения приватных полей **Name** и **Sportsmen**.

Первый конструктор должен принимать название группы и инициализировать массив спортсменов. Второй конструктор должен принимать в себя группу и копировать её название и всех её спортсменов в свой массив спортсменов. Создать три метода **public void Add**, которые позволяют добавить одного и несколько спортсменов или спортсменов другой группы в текущую группу соответственно. Создать метод для сортировки массива участников группы **public void Sort()** в порядке возрастания их времени. Создать метод **public static Group Merge(Group group1, Group group2)** для слияния двух групп в одну с сохранением упорядоченности. Название соединенной группы должно быть “Финалисты”. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Task5. Японская радиокомпания провела опрос радиослушателей по трем вопросам:

1) Какое животное Вы связываете с Японией и японцами?

2) Какая черта характера присуща японцам больше всего?

3) Какой неодушевленный предмет или понятие Вы связываете с Японией?

Большинство опрошенных прислали ответы на все или часть вопросов. Составить программу получения первых пяти наиболее часто встречающихся ответов по каждому вопросу.

Создать публичную структуру **Response** с приватными полями для хранения ответов на каждый из вопросов. Создать публичные свойства только для чтения приватных полей: **Animal, CharacterTrait, Concept**.

Конструктор должен принимать 3 строковых поля для записи ответов на каждый из вопросов. Создать метод **public int CountVotes(Response[] responses, int questionNumber)**, который считает, сколько ответов на выбранный вопрос (от 1 до 3) из списка анкет совпадает с текущей анкетой. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.

Создать публичную структуру **Research** с приватными полями названия и массива ответов радиослушателей. Создать публичные свойства только для чтения приватных полей: **Name, Responses**.

Конструктор должен принимать название и инициализировать массив ответов. Создать метод **public void Add(string[] answers)**, который добавляет новый ответ на опрос в массив. Создать метод **public string[] GetTopResponses(int question)**, который возвращает до 5 наиболее часто встречающихся ответов по указанному вопросу. Создать публичный метод **Print()** для вывода информации о необходимых полях структуры.