**Общие положения**

***Общие положения Практикума по программированию.***

За семестр будет 4 задания, каждое оценивается в 10 баллов. За просрочку сдачи (сдача после занятия-дедлайна) штраф -2 балла за каждое просроченное занятие, но максимальный штраф за просрочку -6 баллов (т.е. за задание студент получит 4 балла).

Списывание карается штрафом. Тотальное списывание – 0 баллов за задачу без права пересдачи. Локальная неспособность объяснить происходящее тоже штрафуется. Неполный функционал задачи при отсутствии желания доделывать – не полный балл за задачу. Тут главное каждому преподавателю установить единые критерии для своих групп. Позже возможно выработаем общие критерии.

Шкала:

35 баллов и выше – автомат без участия в зачете;

30-34 балла – зачет в виде собеседования (неформальный автомат);

Остальные приходят на зачет с целью получить итоговую сумму не менее 51 балла. На зачете будут решать задачи, которые выдавались в семестре, но с нуля и чужие варианты, при этом бальная оценка задач существенно возрастет (ориентировочно полная задача – 30 баллов). Т.е. для большинства это будет очень сложно. Цель: основной путь получения зачета своевременная сдача задач в течение семестра. Предполагается, что практикум станет точкой раннего отсева студентов неспособных/не желающих программировать. Я заранее уведомлю об этом деканат. Так что прошу точно вести записи о достижениях студентов, чтобы студенты к нам потом не придирались.

**Задание 1**

Т.к. решение первой задачи требует умения работать с циклами и списками прошу назначать дедлайн по первому заданию не ранее 3го занятия компьютерного практикума.

Для многих решение задачи будет выглядеть сложным. В связи с этим рекомендую показать, что решение задачи разбивается на несколько подзадач. Например, по первому заданию: получение и обработка ввода пользователя; отрисовка игрового поля; проверка условий и т.п. Можно предложить еще до сдачи полной задачи защитить отдельные подзадачи.

Задачи со звездочкой выдаются студентам только по желанию и одно и тоже задание со звездочкой выдается не более чем 2м студентам в группе. Я буду участвовать в приемке задания со звёздочкой (об этом нужно сразу им сообщить). Не нужно сильно запугивать, моя цель – не завалить их, а познакомиться с активом нашего потока.

**Задачи для первого задания Практикума по программированию.**

1. Реализовать программу, с которой можно играть в логическую игру **«Быки и коровы»** (описание правил игры: <http://робомозг.рф/Articles/BullsAndCowsRules> ). Программа загадывает число, пользователь вводит очередной вариант отгадываемого числа, программа возвращает количество быков и коров и в случае выигрыша игрока сообщает о победе и завершается. Сама программа НЕ ходит, т.е. не пытается отгадать число загаданное игроком.

Взаимодействие с программой производится через консоль, при запросе данных от пользователя программа сообщает, что ожидает от пользователя и проверяет корректность ввода.

1. Реализовать программу, при помощи которой 2 игрока могут играть в **«Крестики-нолики»** на поле 3 на 3. Взаимодействие с программой производится через консоль. Игровое поле изображается в виде трех текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты новой отметки на поле) и проверяет корректность ввода. Программа должна уметь автоматически определять, что партия окончена, и сообщать о победе одного из игроков или о ничьей. Сама программа НЕ ходит, т.е. не пытается ставить крестики и нолики с целью заполнить линию.
2. Реализовать программу, при помощи которой 2 игрока могут играть в игру **«Супер ним»**. Правила игры следующие. На шахматной доске в некоторых клетках случайно разбросаны фишки или пуговицы. Игроки ходят по очереди. За один ход можно снять все фишки с какой-либо горизонтали или вертикали, на которой они есть. Выигрывает тот, кто заберет последние фишки. (описание правил игры: <https://www.iqfun.ru/articles/super-nim.shtml> )

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты новой отметки на поле) и проверяет корректность ввода. Программа должна уметь автоматически определять, что партия окончена, и сообщать о победе одного из игроков. Сама программа НЕ ходит, т.е. не пытается выбирать строки или столбцы с целью победить в игре.

1. Реализовать программу, с которой можно играть в игру **«19»**. Правила игры следующие. Нужно выписать подряд числа от 1 до 19: в строчку до 9, а потом начать следующую строку, в каждой клетке по 1 цифре (не числу (см пример по ссылке)). Затем игроку необходимо вычеркнуть парные цифры или дающие в сумме 10. Условие - пары должны находиться рядом или через зачеркнутые цифры по горизонтали или по вертикали. После того как все возможные пары вычеркнуты, оставшиеся цифры переписываются в конец таблицы. Цель - полностью вычеркнуть все цифры. (описание правил игры: <http://podelki-fox.ru/igry-dlya-detey-na-bumage-s-chislami/> )

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде трех текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять, что нужно выписать новые строки с цифрами и то, что партия окончена. Сама программа НЕ ходит, т.е. не пытается выбирать пары цифр с целью окончить игру.

1. Реализовать программу, при помощи которой 3 игрока могут играть в игру **«Лоскутное одеяло»**. Правила игры следующие. На поле, имеющем размер 4 на 5 клеток за один ход каждый игрок должен заполнить одну клетку своим символом. Игрок старается, чтобы его символы были как можно дальше друг от друга. В ходе игры ведется подсчет очков: за каждое соседство клеток с одинаковыми символами игроку, владельцу символа добавляется одно штрафное очко. Соседними считаются клетки, имеющие общую сторону или расположенные наискосок друг от друга. Выигрывает тот, у кого в конце игры меньше всего штрафных очков.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 4 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять количество штрафных очков и окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается заполнять клетки символами с целью выиграть игру.

1. Реализовать программу, при помощи которой 2 игрока могут играть в игру **«Клондайк»**. Правила игры следующие. Игра ведётся на игровом поле размером 10 на 10 клеток. Игроки по очереди выставляют в любую свободную клетку по отметке, и тот игрок, после чьего хода получилась цепочка длиной хотя бы в 3 отметке, проигрывает. При этом в цепочке считаются как свои отметки, так и отметки соперника, у игровых фишек как бы нет хозяина. Цепочка - это ряд фишек, следующая фишка в котором примыкает к предыдущей с любого из 8-ми направлений. (описание правил игры: <https://www.iqfun.ru/printable-puzzles/klondike-igra.shtml> **)**

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 10 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается ставить в клетки отметки с целью выиграть игру.

1. Реализовать программу , при помощи которой 2 игрока могут играть в игру **«Максит»**. Правила игры следующие. В клетках квадрата 3 на 3 пишутся случайные числа из диапазона от 1 до 9. Начинающий выбирает любое понравившееся ему число и вычеркивает его, прибавляя к своей сумме. Второй игрок может выбрать любое из оставшихся чисел того столбца, в котором первый игрок делал свой предыдущий ход. Он тоже вычеркивает выбранное число, прибавляя его к своей сумме. Первый игрок далее поступает аналогично, выбирая число-кандидата из той строки, в которой второй игрок ходил перед этим. Может так случиться, что у какого-то игрока не будет хода. Тогда его соперник продолжает игру, делая ход в той же строке (для первого игрока) или в том же столбце (для второго игрока), что и до этого. Игра заканчивается, когда оба играющих не имеют ходов. Результат определяется по набранным суммам, у кого она больше, тот и выиграл. При равенстве сумм фиксируется ничья. (описание правил игры: <https://www.iqfun.ru/articles/maxit.shtml> ).

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 3 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять сумму очков каждого из игроков и окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается вычеркивать числа с целью выиграть игру.

1. (\*) Реализовать программу, при помощи которой 2 игрока могут играть в игру **«Мостики»**. Правила игры следующие. В ходе игры каждый из игроков старается построить мост с одного своего берега на другой по камням, образующим массив 4 на 5 (4 камня вдоль берега игрока и 5 камней между берегами). У первого игрока - крестики в качестве камней и берега крестиков (левый и правый край поля), у второго игрока – нолики и берега ноликов (верхний и нижний край поля). Игру можно начинать в любой точке поля. За один ход игрок может соединить два своих соседних камня вертикальным или горизонтальным мостиком (обозначаются в текстовом режиме символами «-» и «|»). Мосты первого и второго игрока пересекаться не должны. Выигрывает тот, кто построит непрерывный мост с одного своего берега на другой. (описание правил игры: <https://www.7ya.ru/article/Chem-zanyat-rebenka-13-igr-na-liste-bumagi-so-slovami-kartinkami/> )

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 9 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять недопустимые ходы (приводящие к пересечению мостов соперников) и окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается строить мосты с целью выиграть игру.

1. (\*) Реализовать программу, с которой можно играть в игру **«Морской бой»**. Программа автоматически случайно расставляет на поле размером 10 на 10 клеток: 4 1-палубных корабля, 3 2-палубных корабля, 2 3-палубных корабля и 1 4-х палубный. Между любыми двумя кораблями по горизонтали и вертикали должна быть как минимум 1 незанятая клетка. Программа позволяет игроку ходить, производя выстрелы. Сама программа НЕ ходит т.е. не пытается топить корабли расставленные игроком.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 10 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты очередного «выстрела») и проверяет корректность ввода. Программа должна уметь автоматически определять потопление корабля и окончание партии и сообщать об этих событиях.

1. (\*) Реализовать программу, с которой можно играть в игру **«Пятнашки»**. Правила игры следующие.

Головоломка представляет собой 15 квадратных костяшек с числами от 1 до 15. Все костяшки заключены в квадратную коробку (поле) размером 4 на 4. При размещении костяшек в коробке остается одно пустое место, которое можно использовать для перемещения костяшек внутри коробки. Цель игры - упорядочить размещение чисел в коробке, разместив их по возрастанию слева направо и сверху вниз, начиная с костяшки с номером 1 в левом верхнем углу и заканчивая пустым местом в правом нижнем углу коробки.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 4 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна считать количество сделанных ходов, уметь автоматически определять недопустимые ходы, окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается упорядочить костяшки с целью выиграть игру.

**Задание 2**

**Задачи для второго задания Практикума по программированию. Общая тема задания «текстовый калькулятор».**

Базовая часть (выполняется всеми самостоятельно!):

Написать калькулятор для строковых выражений вида '<число> <операция> <число>', где <число> - не отрицательное целое число меньшее 100, записанное словами, например "тридцать четыре", <арифмитическая операция> - одна из операций "плюс", "минус", "умножить". Результат выполнения операции вернуть в виде текстового представления числа. Пример calc("двадцать пять плюс тринадцать") -> "тридцать восемь"

Оформить калькулятор в виде функции, которая принимает на вход строку и возвращает строку.

**Дополнительные задания:**

1. Реализовать поддержку операции деления и остатка от деления и работу с дробными числами (десятичными дробями). Пример: calc("сорок один и тридцать одна сотая разделить на семнадцать") -> "два и сорок три сотых". Обрабатывать дробную часть до тысячных включительно, если при делении получаются числа с меньшей дробной частью выполнять округление до тысячных.

**Сложность 2**

1. *Расширение задания 1.* Реализовать поддержку десятичной дробной части до миллионных долей включительно. Реализовать корректный вывод информации о периодической десятичной дроби (период дроби вплоть до 4х десятичных знаков). Пример: calc("девятнадцать и восемьдесят две сотых разделить на девяносто девять") -> "ноль и двадцать сотых и ноль два в периоде ".

**Сложность 3**

1. Реализовать текстовый калькулятор для выражения из произвольного количества операций с учетом приоритета операций. Пример: calc("пять плюс два умножить на три минус один") -> "десять". (Для реализации рекомендуется использовать алгоритмы основанные на польской инверсной записи см. например, <https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C> )

**Сложность 3**

1. *Расширение задания 3.* Добавить поддержку приоритета операций с помощью скобок. Пример: calc("скобка открывается пять плюс два скобка закрывается умножить на три минус один") -> "двадцать".

**Сложность 3**

1. Добавить возможность использования отрицательных чисел. Пример: calc("пять минус минус один") -> "шесть".

**Сложность 1**

1. Добавить возможность оперировать с дробями (правильными и смешанными). Реализовать поддержку сложения, вычитания и умножения, дробей. Результат операций не должен представлять неправильную дробь, такие результаты нужно превращать в смешанные дроби. Пример: calc("один и четыре пятых плюс шесть седьмых ") -> "два и двадцать три тридцать пятых".

**Сложность 3**

1. *Расширение задания 6.* Добавить автоматическое сокращение дроби в ответе. Пример: calc("одна шестая умножить на две третьих") -> "одна девятая".

**Сложность 1**

1. *Расширение задания 1.* Добавить операции возведения в степень и тригонометрические операции синус, косинус, тангенс и константу пи. Допускается как минимум одна из этих функций в выражении с обычными операциями. Пример: calc("два в степени четыре") -> "шестнадцать". Пример: calc("синус от пи разделить на четыре") -> "ноли и семьсот семь тысячных".

**Сложность 1 или 2**

1. Добавить комбинаторные операции перестановки, размещения и сочетания. Пример: calc("размещений из трех по два") -> "шесть".

**Сложность 1 или 2**

1. Диагностировать ошибки: неправильную запись числа; неправильную последовательность чисел и операций; (задание 1) деление на ноль; (задание 3) неправильную последовательность чисел и операций; (задание 4) некорректный баланс и вложенность скобок; (задание 6) некорректную запись числа

**Сложность 1 или 2**

**Для каждого студента формируется комплексное задание из сочетания пунктов. Суммарная сложность комплексного задания должна быть не менее 7. Все выбранные пункты должны работать в функции calc() одновременно.**

**Приветствуется выполнение заданий с суммарной сложностью более 7 (рекомендуется отмечать такие решения дополнительными баллами). Если реализовано 2 и более пунктов сложностью 3, то, считается, что выполнено задание со звездочкой.**

**Сложность пункта с интервальной сложностью (например: Сложность 1 или 2) определяется преподавателем в зависимости от сочетания этого пункта с другими пунктами, выбранными студентом. В случае неоднозначности сочетания требований двух пунктов студент предлагает преподавателю свое видение итоговой постановки задачи, а преподаватель согласует его (рекомендуется фиксировать договоренность в письменном виде). Преподаватель имеет право увеличить оценку сложности комплексного задания в случае существенного увеличения трудоемкости при сочетании двух пунктов.**

**Преподаватель имеет право добавить пункты с собственной постановкой задачи и оценкой сложности. Студент может предложить свои пункты и реализовывать их в случае предварительно согласования с преподавателем постановки задачи и оценки сложности пункта. Рекомендуется фиксировать дополнительные пункты письменно. Желательно, чтобы дополнительные пункты составляли не более половины суммарной трудоемкости комплексного задания.**