Задача A. "Перетворення рядка"

Ідея розв'язання.

Потрібно знайти, яка з букв зустрічається найчастіше. Всі інші букви потрібно буде замінити нею. Для знаходження кількості повторів кожної букви визначимо масив, елементами якого буде кількість повторів кожної букви, а індексами – самі букви.

Після обрахунку кількості повторів букв у рядку, визначимо максимальну величину повторів. Залишиться відняти від загальної кількості букв у рядку максимальну величину повторів певної букви.

Задача B. "Сума цифр"

Ідея розв'язання.

У зв’язку з тим, що число може бути дуже довгим, вводимо його як рядок. Якщо від коду символа-цифри відняти код нуля, то отримаємо число, що дорівнює значенню цифри. Так знаходимо суму всіх цифр числа. Збільшуємо кількість сумувань. Після цього переводимо отриману суму назад у рядок, використовуючи такий самий підхід. Повторюємо ці операції до тих пір, поки довжина отриманого рядка не стане дорівнювати одиниці.

Задача C. "Колода карт"

Ідея розв'язання.

В даній задачі було задане просте правило для "випадкової" перестановки. Насправді, така перестановка не є випадковою, і ми це можемо довести.

Візьмемо просту колоду карт - "1 2 3 4 5", і виконаємо один раз перестановку. Після цього ми отримаємо таку перестановку: "5 3 1 4 2". Якщо прочитати правило перестановки, то ми помітимо, що перестановка ніяк не залежить від карти, яка знаходиться на певному місці. Тобто, перша карта завжди стане на третє місце, якщо карт всього 5. Потім вона стане на друге місце (бо вона стала на третє, а після одної перестановки 3 тепер на другому місці), потім на п’яте, і знову на перше.

Можна замітити, що таких перестановок буде не більше чим n. Якщо ми отримали перестановку, яка була вже раніше, і через те, що перестановка не

залежить від кількості попередніх перестановок, ми можемо взяти остачу від ділення k на кількість перестановок, які ми виконали.

Тому, спочатку нам потрібно перемішувати карти до тих пір, поки ми не отримаємо початкову перестановку. Потрібно підраховувати скільки разів ми виконали перестановку, щоб отримати початкову. Назвемо це число cTrys.

Після цього ми повинні виконати перестановку стільки разів, скільки залишиться в залишку після ділення k на cTrys.

Задача D. "Дивна Фібоначчі"

Ідея розв'язання.

В даній задачі потрібно було згенерувати числа по даній послідовності. Але в даній послідовності була одна проблема – те, що вона збільшувалась в своїх розмірах дуже швидко.

При цьому в задачі обмеження кількості чисел в даній послідовності були дані не явно. Тому, потрібно було рішити дві задачі: Коли зупинитись, і як зменшити розмір послідовності.

Обидві підзадачі рішались дуже просто: якщо в результаті ми отримали число, більше за те, яке було відоме, ми опрацьовували його як нескінченне.

В результаті цього можемо замітити, що може йти дуже багато таких нескінченних чисел підряд. А знаючи базове правило математики, що коли ми додаємо два додатних числа, то ми отримаємо ще більше число. Тому такі числа можна відкинути від наших розрахунків. Потрібно при цьому не порушити послідовність самих чисел. Це можна зробити за допомогою черги, або способом наведеним нижче.

Даний спосіб відкидає два останніх числа, якщо в результаті суми ми отримаємо два таких нескінченних числа підряд. Хоч між двома нескінченними числами стоїть якесь менше число, але з обох боків ми вже отримали числа які більше за k, тому ми його можемо спокійно відкинути.

Задача E. "Метріксополія"

Ідея розв'язання.

В даній задачі потрібно було зробити частинку тетрісу. Кожен раз розраховувати відстань від самого верху до низу, де вже є кімната, не

оптимально через розміри поля. І таке рішення не набирало максимальну кількість балів.

Для рішення даної проблеми потрібно було зберігати висоту в додатковому масиві height. Після того, як ми скидаємо базовий блок зверху, ми можемо розрахувати координати, на які впаде той чи інший блок.

Спочатку рішимо більш просту задачу: У нас є базовий блок, ширина якого 1 і перша кімната на відстані minHeight від низу. (Примітка: Такого тесту не було в самій задачі, хоча цього і не було сказано в умові. Була надана спрощена умова без даної примітки, через те, що був форум, де всі могли задати питання щодо розуміння умови).

Після цього, маючи дані з масиву height, ми можемо взнати де ця сама нижня кімната потім опиниться. Запишемо в змінну yDiff різницю між height і minHeight. Тепер перша кімната опиниться в координатах x, yDiff.

Для того, щоб скинути більш широкий базовий блок, нам потрібно знайти максимальне yDiff, щоб кімната не опинилась в іншій.

Тепер, ми оновлюємо масив height новими значеннями, які ми розраховуємо з yDiff, і відстані від верху базового блоку до самої верхньої кімнати в даному блоці.

В наведеному рішенні, minHeight а також maxHeight розраховувались для базового блоку лише один раз. Але для отримання максимальної кількості балів це не обов’язково.