1 Problem 6.1

Додаємо окремо до M_1 стрічку та використовуємо її для двійкового лічильника на основі вхідного слова.

2 Problem 6.2

- Для додавання використаємо двійкову систему із відповідним алгоритмом додавання чисел із бітом переносу, тобто за властивістю двійкової системи будемо мати логарифмічну складність по пам'яті.

  Для множення використовуємо знову ж таки двійкову систему. Застосовуємо алгоритм множення в стовпчик. Маємо логарифмічну складність по пам'яті.

- Загалом, перевірка однаковості рядків не вимагає додаткової пам'яті, лише декілька станів.

- Будемо використовувати додаткову стрічку для запам'ятовування позиції у рядку y. І послідовно перевіряти однаковість символів по довжині $|x|$ починаючи з цієї позиції, записаної на додатковій стрічці. Будемо інкрементувати лічильник на додатковій стрічці при 'поразці' перевірки на певній позиції. При успіху МТ буде повертати 1, якщо всі позиції пройдено (лічильник досяг значення $|y| - |x| + 1$), то повертаємо 0.

3 Problem 6.4

Рахуємо послідовно нулі, далі рахуємо одиниці, при чому при знаходженні нуля між одиницями повертаємо 0 одразу. Далі потрібно порівняти обидва числа (кількість нулів та одиниць). Повертаємо 1, якщо вони збігаються. Маємо логарифмічну складність по пам'яті.

4 Problem 6.5

Використовуємо двійковий лічильник із підтримкою від'ємних чисел. При значенні лічильника менше за 0 ми повинні повернути 0, оскільки ця ситуація можлива лише при неправильній дужковій структурі. При зчитуванні '(' додаємо до лічильника 1, при ')' віднімаємо одиницю. При використанні двійкової системи маємо логарифмічну складність по пам'яті.

5 Problem 6.6

Будуємо аналогічну систему попередній задачі, окрім:

- Маємо різні види дужок, тому встановлюємо кожен лічильник для окремого типу на окремій стрічці.
- Можуть бути такі недопустимі ситуації — $(]$. Це можна загалодити, використовуючи ще одну окрему стрічку для запам'ятовування поточного відкритого типу дужок.

Використовуємо двійкові лічильники та константну кількість комірок для запам'ятовування типу. Отже маємо логарифмічну складність по пам'яті.