I. Необходимо доказать, что МНР-программа

1. T(1,0)
2. J(2,3,0)
3. S(0)
4. S(3)
5. J(0,0,2)

вычисляет сумму двух чисел, которые хранятся в первом и втором регистрах.

Доказательство будем проводить путем пошагового построения таблицы конфигураций, соответсвующей заданной программе.

Начальная конфигурация:

0	1	2	3	4
0	x	y	0	0

После команды T(1,0) имеем:

0	1	2	3	4
x	x	y	0	0

Далее команда J(2,3,0) сравнивает значение второго регистра (y) со значением третьего (0). Предположим, что они различны, тогда выполняется следующая команда.

После команды S(0) имеем:

0	1	2	3	4
x+1	x	y	0	0

После команды S(3) имеем:

0	1	2	3	4
x+1	x	y	1	0

Далее команда J(0,0,2) сравнивает значение нулевого регистра с собой же, в результате чего повторно выполняется вторая команда.

Команда J(2,3,0) сравнивает значение второго регистра (y) со значением третьего (1). Если они совпадают, значит мы уже вычислили необходимую сумму, и программа завершается. Иначе снова выполняются следующие команды.

После команды S(0) имеем:

0	1	2	3	4
x+2	x	y	1	0

После команды S(3) имеем:

0	1	2	3	4
x+2	x	y	2	0

Выполняется J(0,0,2).
J(2,3,0) снова сравнивает значение второго регистра (y) со значением третьего (2). Если они совпадают, значит мы уже вычислили необходимую сумму, и программа завершается. Иначе программа снова продолжает выполнение.

Так как во втором регистре хранится некоторое натуральное число (y), а значение третьего регистра каждый раз увеличивается на 1, то за конечное число шагов знаения во втором и третьем регистрах будут равны.

Поскольку, каждому шагу увеличения числа в третьем регистре предшествовало увеличение числа в нулевом, то, по итогу, нулевой регистр будет содержать сумму заданных чисел x и y.

II. Предположим, что программа

1. T(1,0)
2. J(2,3,0)
3. S(0)
4. S(3)
5. J(0,0,2)

вычисляет функцию от трех переменных, которые хранятся в первом, втором и соответсвенно третьем регистрах.

Мы «фактически» пошагово увеличиваем начальное значение третьего регистра (z) на 1 до тех пор, пока значение в нем не совпадет со значением во втором регистре (y).

В ситуации, если z изначально было больше чем y — программа не остановится. Иначе же, мы на каждом шаге прибавляли 1 к значению в нулевом регистре (после первой команды там x), до равенства между y и z, что соотвествует прибавлению разницы между ними.

Итого, функция, которую вычисляет программа — это
$f(x, y, z) = x + (y - z)$ при $y \geq z$.

III. Попробуем «упростить смысл» функции $f(x, y) = x \cdot y$
По факту, умножение x на y — это суммирование x-а y раз, или же увелечение y раз нуля на x.

Комментарий: машина, которая лежит на GitHub, записывает переменные, начиная не с первого, а с нулевого регистра. Поэтому для наведения «сахара» я сместил все значения регистров влево с помощью первых трех команд.

f(x, y) можно вычислить с помощью программы

1. T(1,2)
2. T(0,1)
3. Z(0)
4. J(2,4,0)
5. J(1,3,9)
6. S(3)
7. S(0)
8. J(0,0,5)
9. S(4)
10. Z(3)
11. J(0,0,4)

Ее корректность докажем путем пошагового построения таблицы конфигураций.

Начальная конфигурация и результат первых трех команд:

0	1	2	3	4	5
x	y	0	0	0	0
x	y	y	0	0	0
x	x	y	0	0	0
0	x	y	0	0	0

Теперь в первом и втором регистрах лежат x и y соответсвенно. Тогда в третьем регистре будет лежать фактическое увеличение нуля в рамках одного шага умножения, а в четвертом — счетчик шагов умножения.

Команда J(2,4,0) определяет сделали ли мы необходимое количество прибавлений x-a. Если значения во втором регистре (y) и четвертом равны, то вычисление окончено, а иначе выполняются следующие команды программы.

Команда J(1,3,9) определяет окончание увеличения результата на x в рамках шага умножения. Если значение в первом регистре (x) и в третьем равны, то мы переходим на девятую команду, которая увеличивает счетчик шагов умножения в четвертом регистре.

Далее следует обнуление третьего регистра (чтобы мы снова могли увеличивать на x) и возвращение к четвертой команде. Если значения в первом и третьем регистрах были различны, то выполняются следующие команды.

После S(3):

0	1	2	3	4	5
0	x	y	1	0	0

После S(0):

0	1	2	3	4	5
1	x	y	1	0	0

Поскольку x и y — некоторые целые числа, то счетчик шагов умножения (четвертый регистр) рано или поздно достигнет значения во втором регистре (y), и процесс увеличения значения третьего регистра до x также будет конечным.

Заметим, что при каждом обнулении третьего регистра и его увелчиении на единицу в рамках одного шага умножения, нулевой регистр увеличивается на единицу без потери предыдущего значения. Так, в результате нескольких шагов умножения мы и получим итоговое произведение.

IV. f(x, y) = max(x, y)
Процесс вычисления максимального значения из двух — это процесс обратный к вычислению минимального. Если, например, мы будем двигаться по целым числам вдоль оси, то первое из чисел x и y, на которое мы наткнемся, будет минимальным, следовательно второе — максимальным. В программе был использован этот прием.

f(x, y) можно вычислить с помощью программы

1. J(0,2,5)
2. J(1,2,0)
3. S(2)
4. J(0,0,1)
5. T(1,0)

Ее корректность докажем путем пошагового построения таблицы конфигураций.

Начальная конфигурация:

0	1	2	3
x	y	0	0

Команда J(0,2,5) сравнивает значение нулевого регистра (x) со значением второго (0). Предположим, что они не совпадают, тогда выполняется следующая команда.

Команда J(1,2,0) сравнивает значение первого регистра (y) со значением второго (0). Предположим, что они тоже не совпадают, выполняется следующая команда.

После S(2):

0	1	2	3
x	y	1	0

Далее команда J(0,0,1) сравнивает значение нулевого регистра с собой же, в результате чего повторно выполняется первая команда.

Команда J(0,2,5) снова сравнивает значение нулевого регистра (x) со значением второго (1). Если они совпадают, значит при движении вправо по целочисленной оси первым мы встретили x, следовательно y больше, и оно макисмальное. Тогда мы переходим на пятую команду, записываем значение певого регистра (y) в нулевой и, т.к. команд более не осталось, завершаем выполнение программы. Если же значения нулевого и второго регистра опять не совпали, выполняется следующая команда.

Команда J(1,2,0) снова сравнивает значение первого регистра (y) со значением второго (1). Если они совпадают, значит при движении вправо по целочисленной оси первым мы встретили y, следовательно x больше, и оно макисмальное. Т.к. x изначально хранится в нулевом регистре, мы просто завершаем выполнение программы. Если значения первого и второго регистров различны, то мы продолжаем выполнение команд.

Т.к. и x, и y это какие-то целые неотрицательные числа, а во втором регистре с каждым шагом значение увеличивается на единицу, то за конечное число шагов мы рано или поздно достигнем либо x, либо y, которое будет является минимальным из заданной пары.