Question 1:

a)

The user is asked to enter a pupil’s name. Their input, ’Keon’, is stored in the variable *name*.

The user is asked to enter the number of birds counted out of 80. Their input, 70 is stored in the variable *count result*.

This variable, ’70’, *count result* is multiplied by 1.25 and rounded to the nearest whole number. The result, ’88’, is then stored in a variable called *count percentage*.

The sprite then displays the message 88% for 2 seconds.

The program then checks to see if *count percentage* is greater than 85.

The name *Keon* is then added to the list *wren list*.

b)

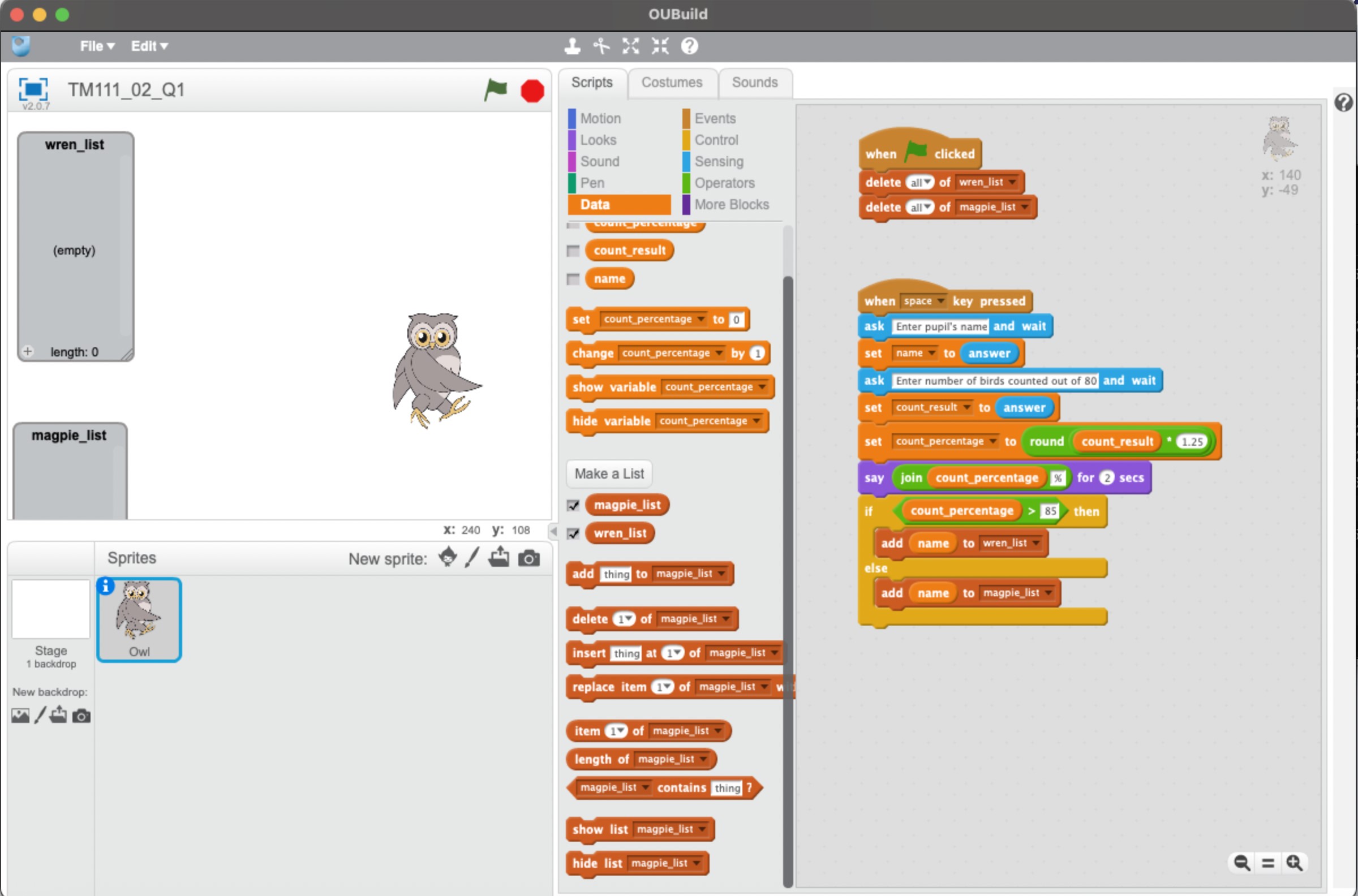
i.

The number 1*.*25 could be stored as a constant.

ii.

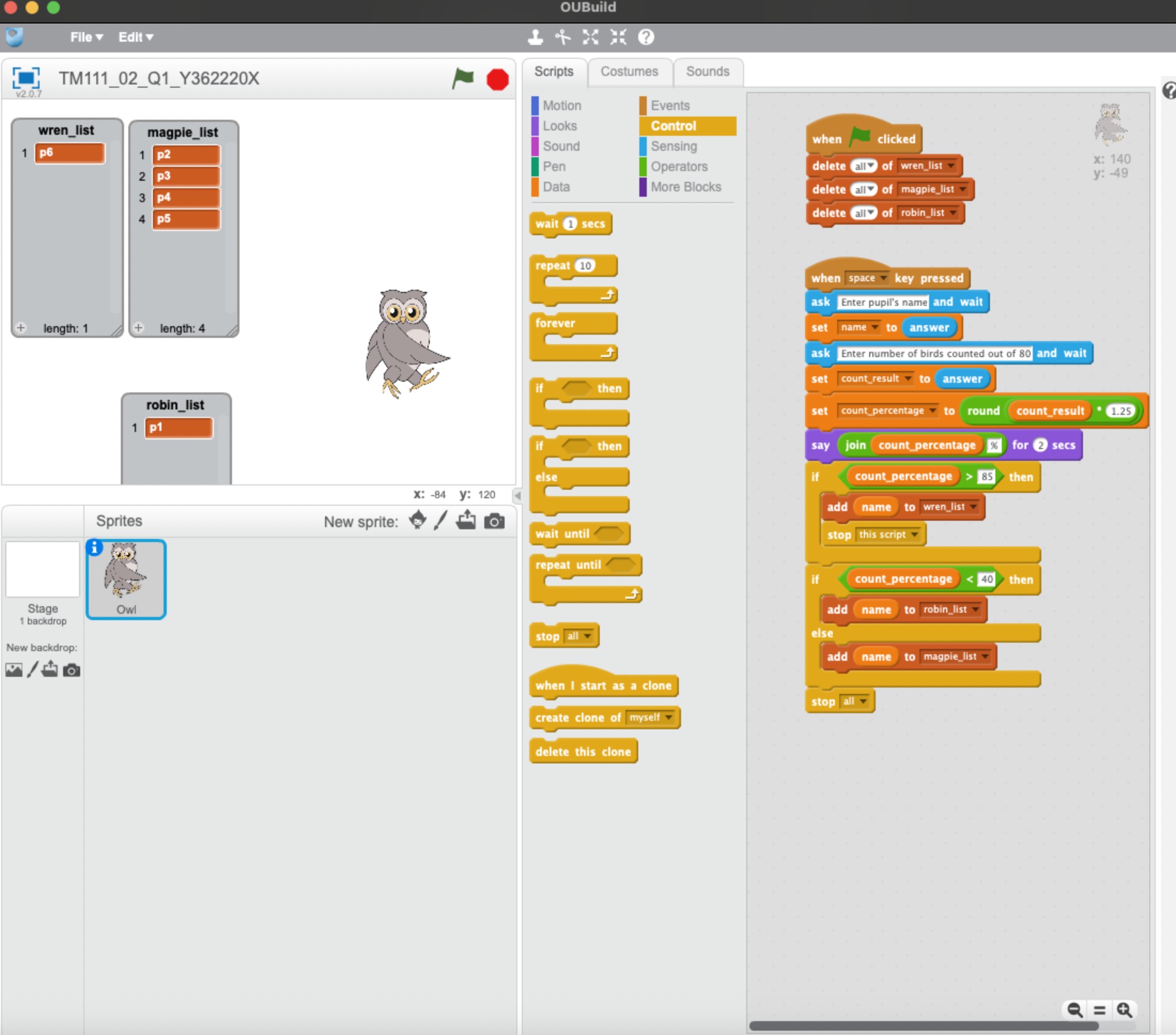
This constant could be called *percentage multiplier*.

c)



d)

i.

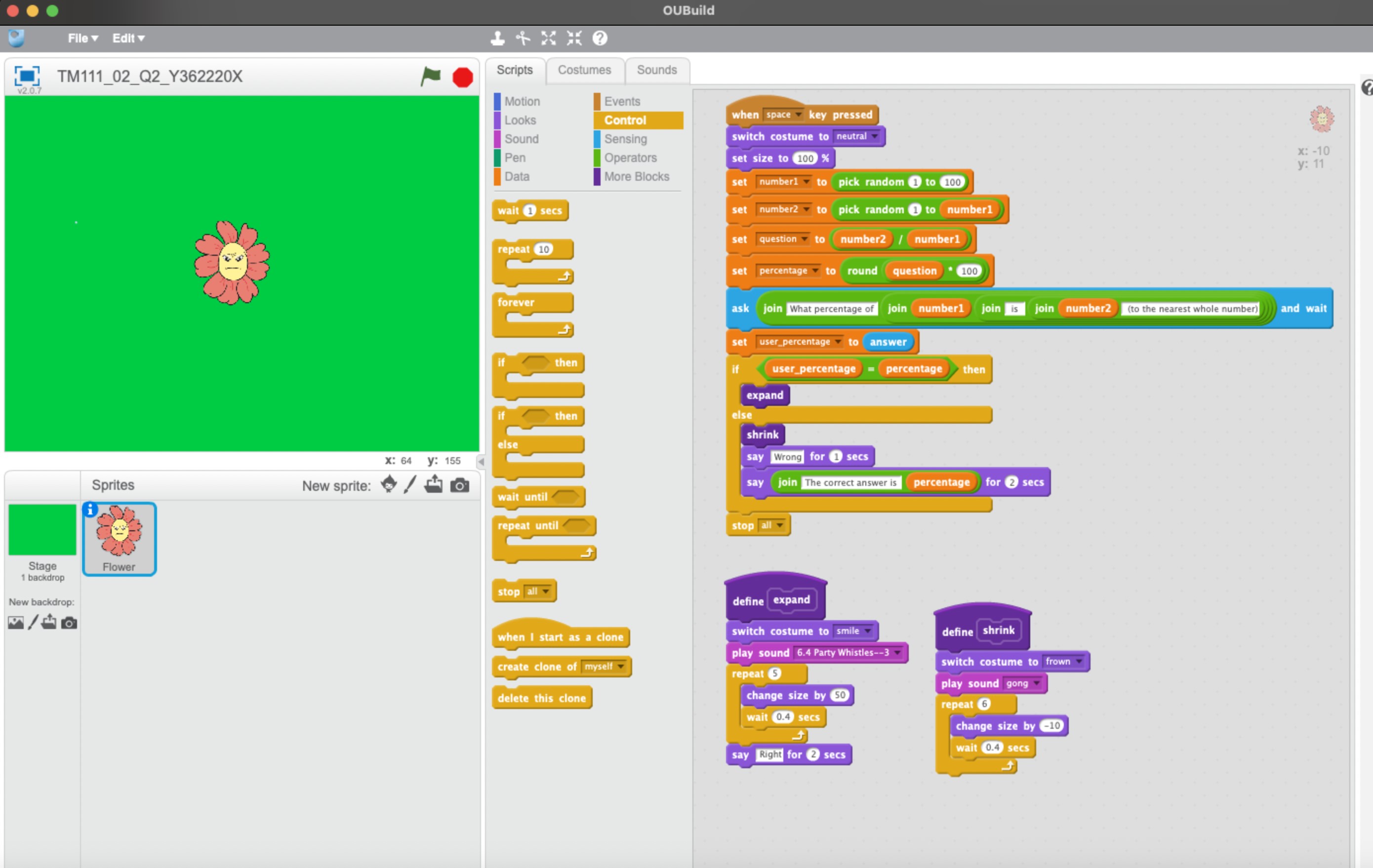


ii.

I could have used three if statements, one for each of the three possible lists.

Question 2:

a)



b)

Question 3:

a)

Clear all variables input word Repeat until end of word

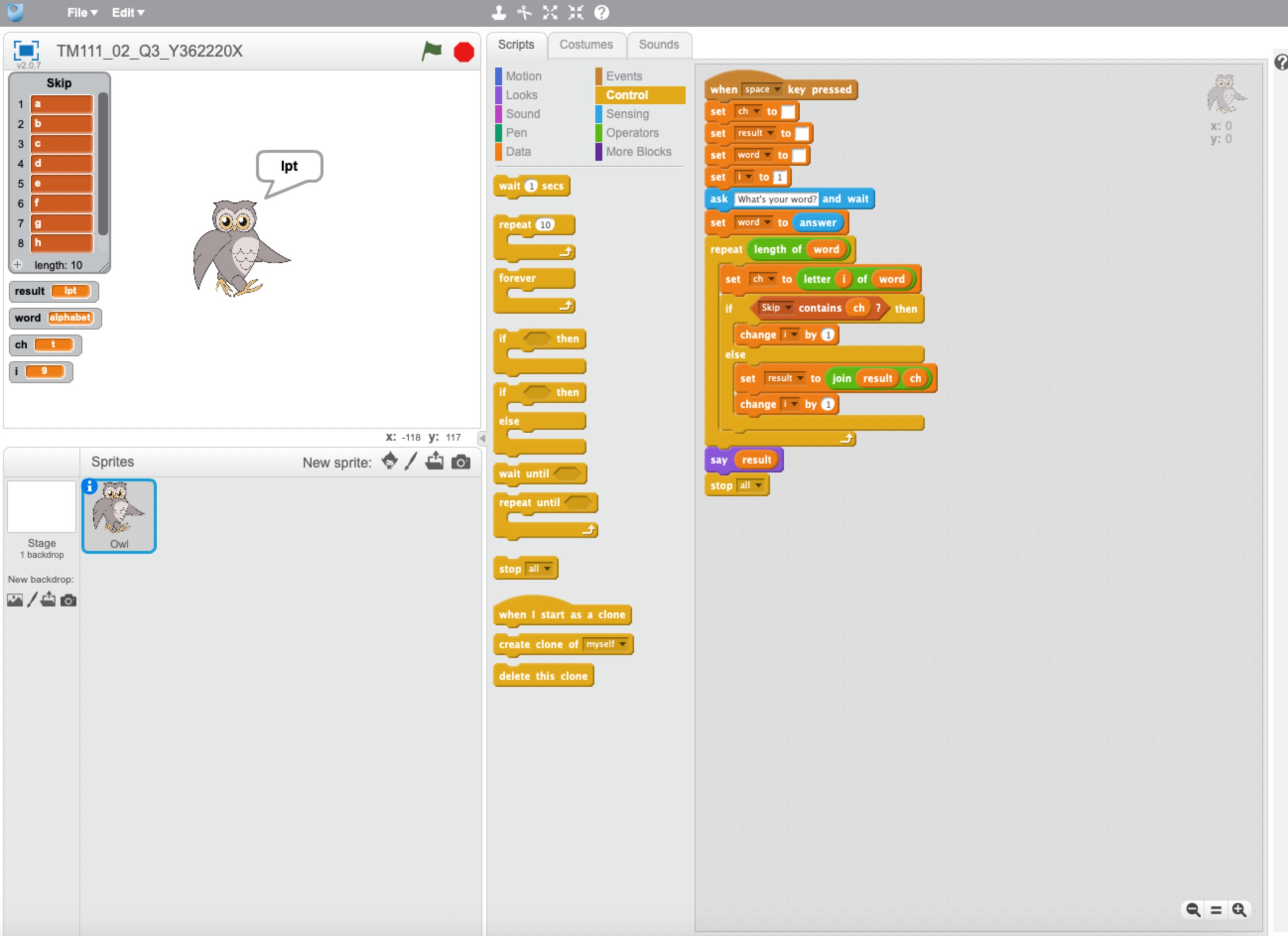
if

check if character *i* is in the banned list.

is on list *i + 1*

else add character to *result* say *result* end

b)



c)

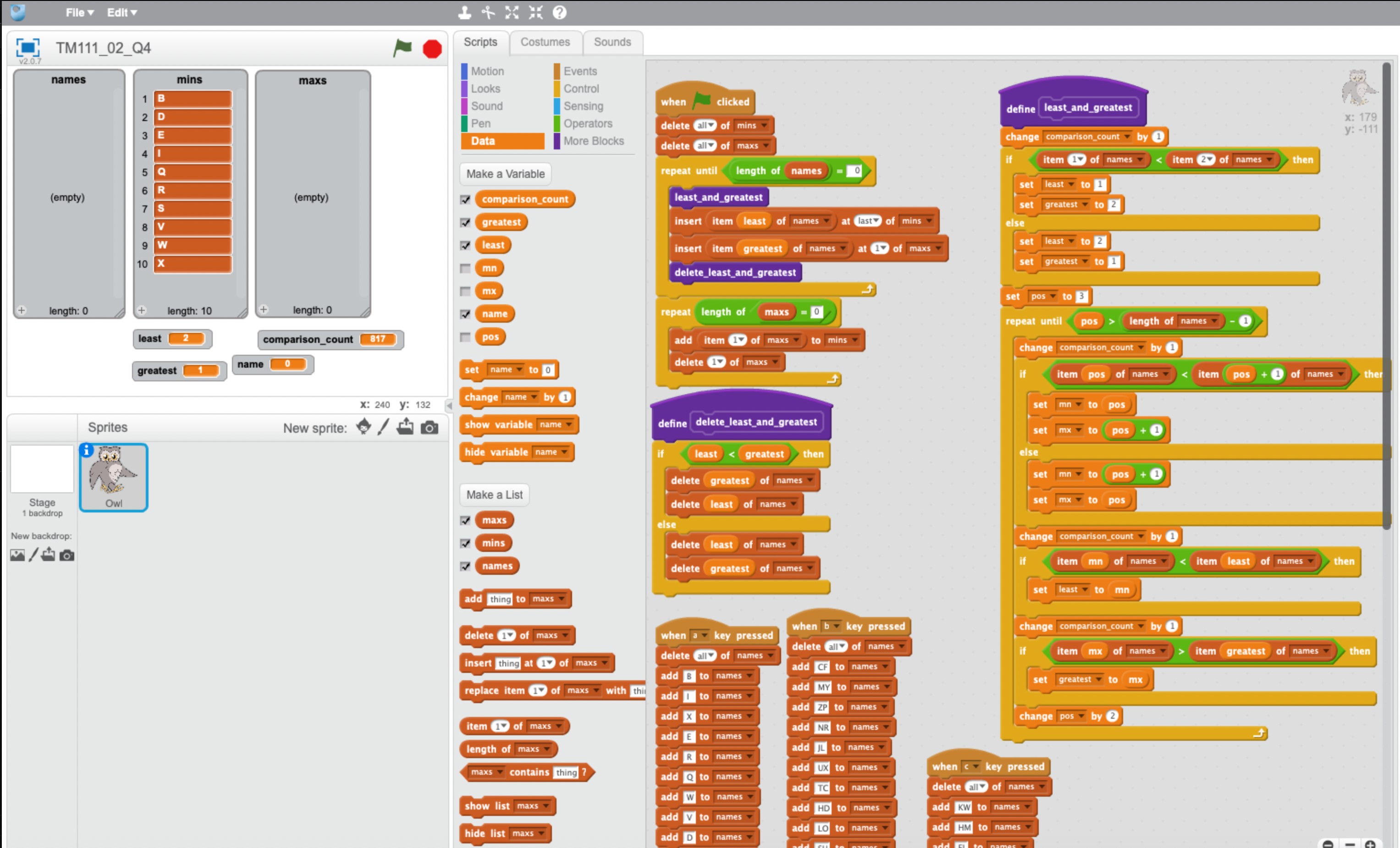
|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Test purpose | input | Expected result |
| 1 | Frist and last letters are in the first ten letters of the alphabet | apple | ppl |
| 2 | Double letters from the first ten letters of the alphabet | keep | kp |
| 3 | Program works despite case of input | BoOkKeEpEr | oOkKpr |

Question 4:

a)

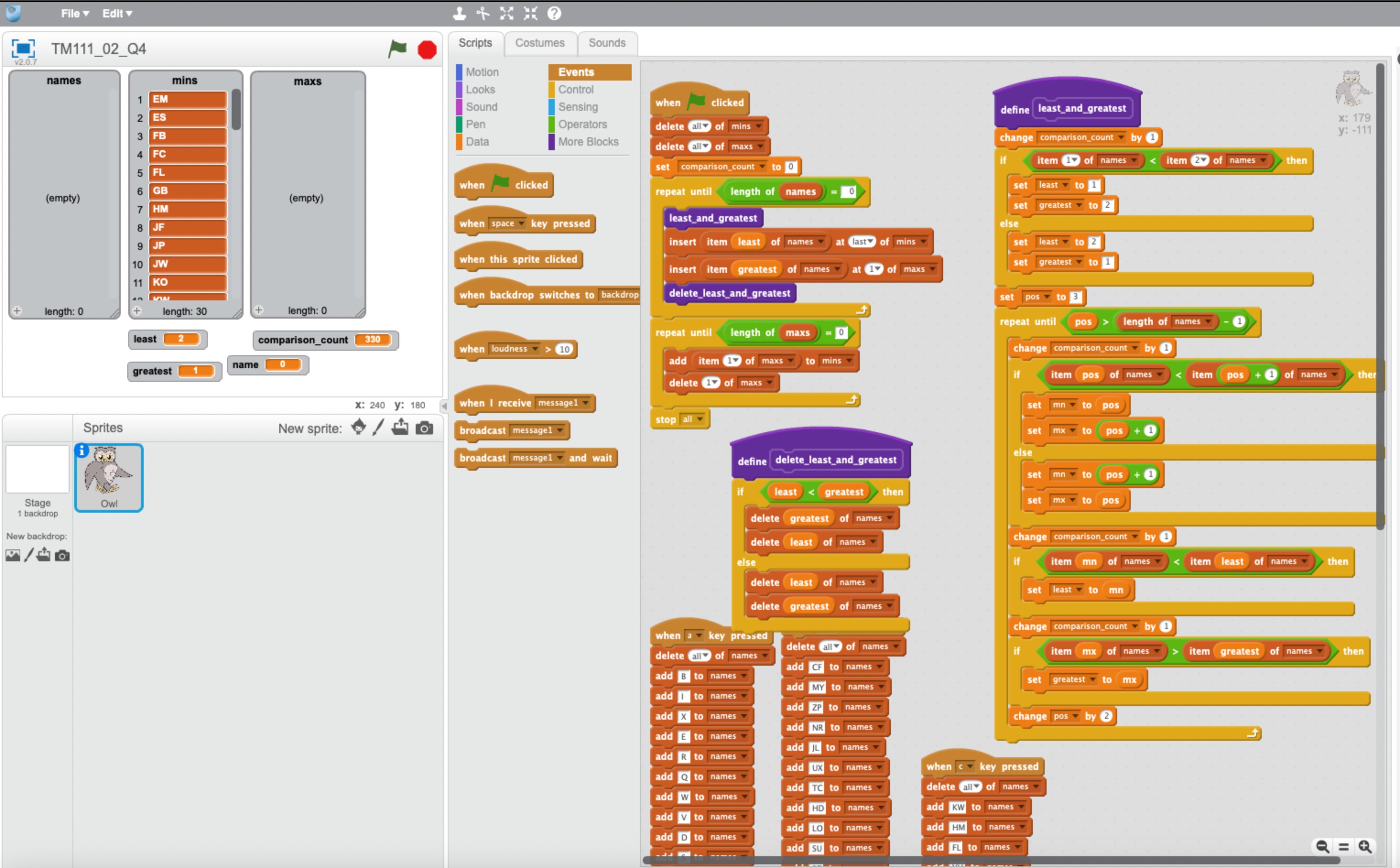
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | Test purpose | Existing data | Expected result | |
|  |  | names | least | greatest |
| 1 | Least name at position 1, greatest at position 3 | B, I, X, E, R, Q, W, V, D, S | 1 | 3 |
| 2 | Least name at position 2, greatest at position 1 | Y, A, C, F, H, J, L, M, O, P | 2 | 1 |
| 3 | Least name at position 1, greatest at position 6 | G, T, K, N, U, Z, H, L, J, O | 1 | 6 |

b)



c)

i.



ii.

|  |  |  |  |
| --- | --- | --- | --- |
| size of list | 10 | 20 | 30 |
| Number of comparisons | 35 | 145 | 330 |

iii.

The number of comparisons is not linear because the number of comparisons does not increases by the same amount each time the list increases by the same amount.

iv.

|  |  |  |  |
| --- | --- | --- | --- |
| size of list (n) | 10 | 20 | 30 |
| *n ×*(*n −*1)2 | 45 | 190 | 435 |
| 0*.*75*× n ×*(*n −*1)2 | 34 | 143 | 327 |

The number of comparisons made is close to the theoretical maximum number of comparisons, which is *n ×*(*n −*1)2, multiplied by 0.75 as program expected a 25% improvement.

Given that the number of comparisons has remained close to the theoretical number for our 3 test, we can assume that this will continue for larger lists. Therefore, if we were sorting a list of 1000 names, the number of comparisons would be approximately

0*.*75*×*1000*×*9992 = 374625.

d)

The double selection sort algorithm would lie approximately in the middle of the selection sort and the bubble sort 1 algorithm.