No sources used

We want to have 4 stacks, one that will deal with the normal arithmetic functions, one that will store the previous input, one that will store the total history of the calculator, and one that will be printed for the history of the calculator

For add, we will push a value, if the stack is zero then we cannot add anything. If the stack is not zero, however, we will pop the top two values of the stack and add them together, storing this into our variable result. We then push result into the stack.

For subtract, we do the same as addition except we subtract the top two values of the stack rather than add them.

For both add and subtract we push a value into the main stack as well as the ‘prev’ stack concurrently. For subtraction, we push the negative of the value. This is so that when we undo we simply pop the top of the main stack and the top of the prev stack and subtract them.

One such example is
add(2) -> stack = {2} prev -> {2}(stack was empty so can’t add)
add(4) -> stack = {6} prev -> {2,4} after popping 4 and 2 to add = 6 and push this result to the main stack

undo() would then pop the main stack and the top of the prev and subtract then push this result onto the main stack again

-> pop 6 (main) and 2(prev) -> 6-2 = 4
->push(4) -> stack = {4} prev->{2}

If we did undo again our stack would be {2} and prev {0}
If we did undo again we throw an empty stack exception.

For re-do we follow the same steps as undo except we add rather than subtract from prev,

For printing, we have a third stack called history that is added upon every function call.. So we have a fourth stack called historyprint so that every time we pop from history, it will push into historyprint and printing this stack will properly portray the history of the calculator.