```
0x24a
--------
assert(0 == msg.value)
$s2 = ad_mask & c[0x4]
m[0x0] = ad_mask & $s2
m[0x20] = 0x2
$s11 = s[sha3(0x0, 0x40)]
$s10 = $m
m[$m] = $s11
$m = $m + (0x20 + (0x20 * $s11))
if ($s11){
  codecopy(0x20 + $s10, codesize(), 0x20 * $s11)
}
m[0x0] = ad_mask & $s2
m[0x20] = 0x2
$s13 = s[sha3(0x0, 0x40)]
m[$m] = $s13
$s12 = $m
$t = $s13
$s14 = $t
$s11 = $t
$m = 0x20 + ($m + (0x20 * $s13))
$t = $s10
$s10 = $s12
$s3 = $t
if ($s11){
  codecopy(0x20 + $s10, codesize(), 0x20 * $s14)
}
m[0x0] = ad_mask & $s2
m[0x20] = 0x2
$s13 = sha3(0x0, 0x40)
$s14 = $s10
$s5 = s[0x3 + $s13]
$s6 = s[0x2 + $s13]
$s7 = s[0x1 + $s13]
$s9 = 0x0
while (0x1) {
  m[0x0] = ad_mask & $s2
  m[0x20] = 0x2
  if ($s9 >= s[sha3(0x0, 0x40)])
        break
  m[0x0] = ad_mask & $s2
  m[0x20] = 0x2
  $s10 = sha3(0x0, 0x40)
  assert($s9 < s[$s10])
  m[0x0] = $s10
  assert($s9 < m[$s3])
  m[0x20 + ($s3 + (0x20 * $s9))] = s[(0x2 * $s9) + sha3(0x0, 0x20)]
  m[0x0] = ad_mask & $s2
  m[0x20] = 0x2
  $s10 = sha3(0x0, 0x40)
  assert($s9 < s[$s10])
  m[0x0] = $s10
  assert($s9 < m[$s14])
  m[0x20 + ($s14 + (0x20 * $s9))] = s[0x1 + ((0x2 * $s9) + sha3(0x0, 0x20))]
  $s9 = 0x1 + $s9
}
if (s[0x8]){
  $s8 = s[0x8]
} else {
  $s8 = block.timestamp
}
$s9 = 0x20 + $m
$s10 = 0x20 + $s9
m[$s10] = $s5
$s10 = 0x20 + $s10
m[$s10] = $s6
$s10 = 0x20 + $s10
m[$s10] = $s7
$s10 = 0x20 + $s10
m[$s10] = $s8
$s10 = 0x20 + $s10
m[$m] = $s10 - $m
m[$s10] = m[$s3]
$s10 = 0x20 + $s10
$s11 = 0x20 + $s3
$s12 = 0x20 * m[$s3]
$s16 = 0x0
while (0x1) {
  if ($s16 >= $s12)
        break
  m[$s16 + $s10] = m[$s16 + $s11]
  $s16 = 0x20 + $s16
}
$s10 = $s12 + $s10
m[$s9] = $s10 - $m
m[$s10] = m[$s14]
$s10 = 0x20 + $s10
$s11 = 0x20 + $s14
$s12 = 0x20 * m[$s14]
$s16 = 0x0
while (0x1) {
  if ($s16 >= $s12)
        break
  m[$s16 + $s10] = m[$s16 + $s11]
  $s16 = 0x20 + $s16
}
return($m, ($s12 + $s10) - $m)
```