```
0x2da
--------
assert(0 == msg.value)
$s2 = c[0x4]
$s4 = intcall7(0xcd5)
assert($s2 < s[0x3])
m[0x0] = 0x3
$s8 = (0x3 * $s2) + sha3(0x0, 0x20)
$s10 = s[$s8]
$s11 = intcall7(0x1cf4)
$s12 = intcall7(0x1cfc)
$s16 = intcall7(0x1d09)
$s18 = $m
if (msize() >= $m){
   $s18 = msize()
}
m[$s18] = 0x20
$m = 0x20 + (0x20 + $s18)
$s12 = $s18
$s13 = 0x0
$s14 = 0x0
while (0x1) {
   if ($s14 >= 0x20)
         break
   $s15 = $s10 * (0x2 ** (0x8 * $s14))
   if (0xff00000000000000000000000000000000000000000000000000000000000000 & $s15){
      assert($s13 < m[$s12])
      mstore8((0x20 + $s12) + $s13, 0x0 byte (0xff00000000000000000000000000000000000000000000000000000000000000 & $s15))
      $t = $s13
      $s13 = $s16
      $s16 = 0x1 + $t
      $t = $s13
      $s13 = $s16
      $s16 = $t
   }
   $t = $s14
   $s14 = $s16
   $s16 = 0x1 + $t
   $t = $s14
   $s14 = $s16
   $s16 = $t
}
$s18 = $m
if (msize() >= $m){
   $s18 = msize()
}
m[$s18] = $s13
$m = 0x20 + ($s18 + (0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffe0 & (0x1f + $s13)))
$s16 = $s18
$s14 = 0x0
while (0x1) {
   if ($s14 >= $s13)
         break
   assert($s14 < m[$s12])
   assert($s14 < m[$s16])
   mstore8((0x20 + $s16) + $s14, 0x0 byte (0xff00000000000000000000000000000000000000000000000000000000000000 & ((m[0x20 + ($s14 + $s12)] >> 0xf8) << 0xf8)))
   $t = $s14
   $s14 = $s16
   $s16 = 0x1 + $t
   $t = $s14
   $s14 = $s16
   $s16 = $t
}
$s10 = s[0x2 + $s8]
m[$m] = $s2
m[0x40 + $m] = s[0x1 + $s8]
m[0x60 + $m] = ad_mask & (ad_mask & $s10)
m[0x80 + $m] = 0xff & ($s10 >> 0xa0)
m[0x20 + $m] = 0xa0
$s8 = 0xa0 + $m
m[$s8] = m[$s16]
$s8 = 0x20 + $s8
$t = m[$s16]
$s9 = 0x20 + $s16
$s14 = 0x0
while (0x1) {
   if ($s14 >= $t)
         break
   m[$s14 + $s8] = m[$s9 + $s14]
   $s14 = 0x20 + $s14
}
$s9 = $t
$t = $s8
$s8 = $s9
$s9 = $s9 + $t
$t = $s8
$s8 = $s9
$s9 = 0x1f & $t
if ($s9){
   $s10 = $s8 - $s9
   m[$s10] = (! ((0x100 ** (0x20 - $s9)) - 0x1)) & m[$s10]
   $s8 = 0x20 + $s10
}
return($m, $s8 - $m)
```