1.a.i) a

Ii) a

Iii) b

b.i) require (block.timestamp <= stop);

ii) Transaction fails and reverts. Throw Out of Gas exception by EVM. State back to the one before the transaction. Still charge the amount before running out of the gas.

C.i) standardization of features, ECR20, transfer, check balance, approve others for transfer.

Fungible tokens are all equivalent and interchangeable with each other, non-fungible tokens are each unique and not interchangeable or replicable. *Your ugly ape is still worthless unfortunately. Unless some other dumbass buys it*

d.i) withdraw(); Balance is set to zero after the transfer function; The attacker can repeatedly call withdraw() in a loop to get tokens because the balance is not equal to 0 before setting it to 0; construct a payable function with a loop of n times, calling withdraw(); gain n\*balance.

ii)

contract Attack {

uint count = 0;

function attack() {

vulnerable.withdrawAll();

function attack(uint n) {

count += 1;

if (count < n) {

attack();

}

}

}

iii) swap the call function and balance setting function in withdraw() function.