Homework 6

> ### Assembly 2
>
> 1. Create a Solidity contract with one function
>   The solidity function should return the amount of ETH that was passed to it, and the function body should be written in assembly
>
> 2. Do you know what this code is doing ?
>
> ```
> push9 0x601e8060093d393df3
> msize
> mstore                      # mem = 000...000 601e8060093d393df3
>                             #     = 000...000 spawned constructor
> payload
>
> # copy the runtime bytecode after the constructor code in mem
> codesize                    # cs
> returndatasize              # 0 cs
> msize                       # 0x20 0 cs
> codecopy                    # mem = 000...000 601e8060093d393df3
> RUNTIME_BYTECODE
>
>
>                             # --- stack ---
> push1 9                     # 9
> codesiz                     # cs 9
> add                         # cs+9 = CS = total codesize in memory
>
> push1 23                    # 23 CS
> returndatasize              # 0 23 CS
> dup3                        # CS 0 23 CS
>
> dup3                        # 23 CS 0 23 CS
> callvalue                   # v 23 CS 0 23 CS
> create                      # addr1 0 23 CS
> pop                         # 0 23 CS
>
> create                      # addr2
> ```

> ```
> selfdestruct
> ```
>
> See gist
> The runtime bytecode for this contract is
>
> ```
> 0x68601e8060093d393df35952383d59396009380160173d828234f050f0ff
> ```
>
> 3. Explain what the following code is doing in the Yul ERC20 contract
>
> ```
>         function allowanceStorageOffset(account, spender) -> offset {
>                 offset := accountToStorageOffset(account)
>                 mstore(0, offset)
>                 mstore(0x20, spender)
>                 offset := keccak256(0, 0x40)
>         }
> ```