Given a smart contract called *auction* as

```
parameter (pair mutez address); # the participant's bid amount and address
storage (pair
            (pair int address) # the number of accepted bids made by
                               participants and the contract owner's address
            (pair mutez address) # the current highest bidder's bid amount and
                                address
         );
code {# (pair parameter storage) : []
       # pair (pair @parameter mutez address) (pair @storage (pair int address) (pair mutez
       address)) : []
       # make sure that the participant has contributed at the minimal price 2 tez
       PUSH mutez 2000000;
       AMOUNT; # get the amount bid that is transferred along with this contract call
       IFCMPGE {} { PUSH string "You did not provide enough tez (the minimal
       price 2 tez)."; FAILWITH; };
       # check that the number of accepted bids has not been exceeded 10
       UNPAIR;
       # pair mutez address : pair (pair int address) (pair mutez address) : []
       SWAP;
       # pair (pair int address) (pair mutez address) : pair mutez address : []
       DUP;
       # pair (pair int address) (pair mutez address) : pair (pair int address) (pair mutez
       address) : pair mutez address : []
       CAR;
       # pair int address : pair (pair int address) (pair mutez address) : pair mutez
       address : []
       CAR;
       # int : pair (pair int address) (pair mutez address) : pair mutez address : []
       DIP { PUSH int 10; };
       # int : int : pair (pair int address) (pair mutez address) : pair mutez address : []
       IFCMPLT {
                  # check if the participant's bid is higher than the current  highest bid
                    SWAP;
                    # pair mutez address : pair (pair int address) (pair mutez address): []
                    DIP { DUP; };
                    # pair mutez address : pair (pair int address) (pair mutez
                    address) : pair (pair int address) (pair mutez address) : []
                    DUP;
                    # pair mutez address : pair mutez address : pair (pair int
                    address) (pair mutez address) : pair (pair int address) (pair mutez
                    address) : []
                    DIP { SWAP; };
                    # pair mutez address : pair (pair int address) (pair mutez
                    address) : pair mutez address : pair (pair int address) (pair mutez
                    address) : []
                    DIP { CDR; };
                    # pair mutez address : pair mutez address : pair mutez address    :
                    pair (pair int address) (pair mutez address) : []
                    CAR;
```

```
# mutez : pair mutez address : pair mutez address : pair (pair int
address) (pair mutez address) : []
DIP { CAR; };
# mutez : mutez : pair mutez address : pair (pair int address) (pair
mutez address) : []
IFCMPGT {
            # If the participant' bid is higher than the
            current highest bid, the contract returns the
            previous bid to its owner and updates the store
            with the new current highest bid amount and the
            bidder’s address

            SWAP;
            # pair (pair int address) (pair mutez address) :
            pair mutez address : []
            UNPAIR;
            # pair int address : pair mutez address : pair
            mutez address : []
            SWAP;
            # pair mutez address : pair int address : pair
            mutez address : []
            UNPAIR;
            # mutez : address : pair int address : pair mutez
            address : []
            SWAP;
            # address : mutez : pair int address : pair mutez
            address : []
            CONTRACT unit;
            # convert the current highest bidder's address to
            a contract
            IF_SOME {} { FAILWITH; };
            # contract unit : mutez : pair int address : pair mutez
                  address : []
            SWAP;
            # mutez : contract unit : pair int address : pair mutez
            address : []
            DUP;
            # mutez : mutez : contract : pair int address : pair
            mutez address : []
            PUSH mutez 0;
            # mutez : mutez : mutez : contract unit : pair int
            address : pair mutez address : []
            IFCMPLT {
                  # mutez : contract unit : pair int address : pair
                        mutez address : []
                  UNIT; # match the contract type
                  # unit : mutez : contract unit : pair int address :
                  pair mutez address : []
                  TRANSFER_TOKENS; # transfer the money to
                  the current highest bidder
                  # operation : pair int address : pair mutez
                  address : []
                  PUSH int 1;
```

```
                # int : operation : pair int address : pair
                mutez address : []
                SWAP;
                # operation : int : pair int address : pair
                mutez address : []
                DIP 2 { UNPAIR; };
                # operation : int : int : address : pair
                mutez address : []
                DIP { ADD; };
                # operation : int : address : pair mutez
                address : []
                DIP { PAIR; };
                # operation : pair int address : pair mutez
                address : []
                DIP { PAIR; };
                # operation : pair (pair int address) (pair
                mutez address) : []
                NIL operation; SWAP; CONS; PAIR;
                # pair (list operation) (pair (pair int
                address) (pair mutez address)) : []
        }
        {
                # mutez : contract unit : pair int address : pair
                mutez address : []
                DROP;
                # contract unit : pair int address : pair mutez
                address : []
                DROP;
                # pair int address : pair mutez address : []
                UNPAIR;
                # int : address : pair mutez address : []
                PUSH int 1;
                # int : int : address : pair mutez address : []
                ADD;
                # int : address : pair mutez address : []
                PAIR;
                # pair int address : pair mutez address : []
                PAIR;
                # pair (pair int address) (pair mutez
                address) : []
                NIL operation; PAIR;
                # pair (list operation) (pair (pair int
                address) (pair mutez address)) : []
        }
}
{       # If the participant' bid is lower than the current
        highest bid, the contract rejects the bid.
        # pair mutez address : pair (pair int address)
        (pair mutez address) : []
        PUSH string "your bid is lower than or equal to
        the current highest bid."; FAILWITH;
```

```
                    };
                }
                {
                    # if the number of accepted bids exceeds 10
                    PUSH string "The auction is closed since the number of
                    accepted bids exceeds 10."; FAILWITH;
                };
            };
```

- black text is Michelson instructions.

- blue text is comments.

Intuitively, the auction smart contract works according to the following strategy

- The contract manager originates the contract and provides the owner's address.

- The input parameter contains a pair of the bid amount and the bidder's address

```
(pair mutez address)
```

- The storage of the contract contains a pair of

(1) a pair of the number of bids made by participants and the contract owner's address

(2) a pair of the current highest bid amount and the bidder's address

```
(pair
      (pair int address)
      (pair mutez address)
)
```

- The contract will be closed when there are 10 accepted bids (an accepted bid is higher than the previous highest bid).

- A bid is placed by sending money (a bid) to the contract: if the amount of the bid is less or equal than the current highest bid, the contract rejects the bid and this is not considered as an accepted bid. Otherwise, the contract returns the previous bid to its owner and updates the store with the new current highest bid amount and the bidder's address.

1. Given the initial stack as

S :: Pair (Pair 9 "tz1VmefgyE1YPcAa8VurxihsUJXzLUX6HY8Z")
(Pair (Pair 3 "tz1XbA8xcxkPAkB7CwyNq4fnbQdCXunKXHsb")
(Pair 7 "tz1XJu9hk9aYZGqrAWZDsDxMHRoUQ1NwC5CL")) : [ ]

Give the content of the stack after each instruction.

2. The current version does not validate whether the amount that is transferred along with a smart contract call is the same as what is claimed in the input parameter.

Revise the code of the smart contract such that it validates whether these two amounts are the same.

Hint.

- the input parameter is a pair **`(pair mutez address)`**

- the instruction **`AMOUNT`** pushes the amount that is sent along with the contract call.

3. In the current version of the auction contract, when there are 10 accepted bids, the auction is closed as follows:

```
{
      # if the number of accepted bids exceeds 10
      PUSH string "The auction is closed since the number of accepted
      bids exceeds 10."; FAILWITH;
};
```

Revise the code such that it will transfer the current highest bid to the contract owner. (Your code does not have to avoid multiple transfers to the owner. Explain why.)

```
{
      # if the number of accepted bids exceeds 10
      # write the code here
      ...
};
```

4. Each bidder in the auction contract should either win the auction or get the bid back. Discuss whether this statement is true for the given code and propose a solution if it is not.