```
┌──────────────────────────── MODULE TESpec ────────────────────────────┐
EXTENDS Naturals, TLC, FiniteSets, Token

CONSTANTS EXCHANGE,   exchange contract name
          INIT_XTZ    initial (mu)xtz amount

VARIABLES xtzMap,   XTZ amount state of contracts
          orders    orders state

├───────────────────────────────────────────────────────────────────────┤
 some exchange helper operators

Users ≜
  {x ∈ CONTRACTS : x ≠ EXCHANGE}

PickOrder(key) ≜
  LET matches ≜ {x ∈ orders : x.key = key}
  IN   IF matches = {} THEN [xtz ↦ 0, token ↦ 0]
        ELSE  CHOOSE m ∈ matches : TRUE

XTZTransfer(owner, receiver, amount) ≜
  IF owner = receiver
   THEN UNCHANGED xtzMap
   ELSE
  xtzMap′ = [x ∈ CONTRACTS ↦
                 CASE x = owner → xtzMap[x] − amount
                   □   x = receiver → xtzMap[x] + amount
                   □   OTHER  → xtzMap[x]]


├───────────────────────────────────────────────────────────────────────┤
 tez.exchange basic user operators

CreateBuyingOrder(token, buyer, price, xtz_amount) ≜
  LET key ≜ ⟨buyer, token, TRUE, price⟩
      order ≜ PickOrder(key)
      prev_xtz_amount ≜ order.xtz
  IN
   ∧ XTZTransfer(buyer, EXCHANGE, xtz_amount)
   ∧ orders′ = {x ∈ orders : x.key ≠ key} ∪
                {[key ↦ key, xtz ↦ xtz_amount + prev_xtz_amount]}
   ∧ UNCHANGED ⟨tokenMap⟩


ExecuteBuyingOrder(order, executer, token_amount) ≜
  LET token  ≜ order.key[2]
      price  ≜ order.key[4]
      owner  ≜ order.key[1]
```

```
        $consumed\_xtz \triangleq price * token\_amount$
        $remain\_xtz \triangleq order.xtz - consumed\_xtz$
  IN
   $\land remain\_xtz \geq 0$
   $\land XTZTransfer(EXCHANGE, executer, consumed\_xtz)$
   $\land TOKENTransfer(token, executer, owner, token\_amount)$
   $\land orders' =$ IF $remain\_xtz = 0$
                  THEN $\{x \in orders : x.key \neq order.key\}$
                  ELSE $\{x \in orders : x.key \neq order.key\} \cup$
                       $\{[key \mapsto order.key, xtz \mapsto remain\_xtz]\}$

$CreateSellingOrder(token, seller, price, token\_amount) \triangleq$
  LET $key \triangleq \langle seller, token,$ FALSE$, price \rangle$
      $order \triangleq PickOrder(key)$
      $prev\_token\_amount \triangleq order.token$
  IN
   $\land TOKENTransfer(token, seller, EXCHANGE, token\_amount)$
   $\land orders' = \{x \in orders : x.key \neq key\} \cup$
                $\{[key \mapsto key, token \mapsto token\_amount + prev\_token\_amount]\}$
   $\land$ UNCHANGED $\langle xtzMap \rangle$

$ExecuteSellingOrder(order, executer, xtz\_amount) \triangleq$
  LET $token \triangleq order.key[2]$
      $price \triangleq order.key[4]$
      $owner \triangleq order.key[1]$
  IN
   $\land price \neq 0$
   $\land$ LET $consumed\_token \triangleq xtz\_amount \div price$
           $remain\_token \triangleq order.token - consumed\_token$
      IN
       $\land remain\_token \geq 0$
       $\land XTZTransfer(executer, owner, xtz\_amount)$
       $\land TOKENTransfer(token, EXCHANGE, executer, consumed\_token)$
       $\land orders' =$ IF $remain\_token = 0$
                      THEN $\{x \in orders : x.key \neq order.key\}$
                      ELSE $\{x \in orders : x.key \neq order.key\} \cup$
                           $\{[key \mapsto order.key, token \mapsto remain\_token]\}$
```

---

some invariants for checking

```
$xtzMapChecker \triangleq$
  $Sum(Range(xtzMap)) = (Cardinality(CONTRACTS) - 1) * INIT\_XTZ$
```

```
$tokenMapCheckerTE \triangleq$
  $[t \in TOKENS \mapsto Sum(Range(tokenMap[t]))] =$
    $[t \in TOKENS \mapsto (Cardinality(CONTRACTS) - 1) * INIT\_TOKEN]$

$ordersChecker \triangleq$
  $\land\ xtzMap[EXCHANGE] =$
        $Sum(\{\langle order.xtz,\ order.key\rangle : order \in$
                $\{x \in orders : x.key[3] = \text{TRUE}\}\})$

  $\land\ [t \in TOKENS \mapsto tokenMap[t][EXCHANGE]] =$
       $[t \in TOKENS \mapsto$
         $Sum(\{\langle order.token,\ order.key\rangle : order \in$
               $\{x \in orders : x.key[3] = \text{FALSE} \land x.key[2] = t\}\})]$
```

the init behavior

```
$TEInit \triangleq$
  $\land\ xtzMap = [x \in CONTRACTS \mapsto \text{IF } x = EXCHANGE$
                                   $\text{THEN } 0$
                                   $\text{ELSE } INIT\_XTZ]$
  $\land\ tokenMap = [t \in TOKENS \mapsto$
                     $[x \in CONTRACTS \mapsto \text{IF } x = EXCHANGE$
                                        $\text{THEN } 0$
                                        $\text{ELSE } INIT\_TOKEN]]$
  $\land\ orders = \{\}$
  $\land\ pick = [$
        $token \mapsto RandomElement(TOKENS),$
        $user \mapsto RandomElement(Users),$
        $price \mapsto RandomElement(0 \mathrel{..} (INIT\_XTZ \div INIT\_TOKEN))$
     $]$
```

the next behavior

this behavior will pick random token and *executer* to test possible operations

```
$TENext \triangleq$
  $\land\ pick' = [$
        $token \mapsto RandomElement(TOKENS),$
        $user \mapsto RandomElement(Users),$
        $price \mapsto RandomElement(0 \mathrel{..} (INIT\_XTZ \div INIT\_TOKEN))$
     $]$

  $\land\ \lor\ \land\ xtzMap[pick.user] > 0$
         $\land\ \lor\ CreateBuyingOrder(pick.token,$
                  $pick.user,$
```

```
            $pick.price,$
            $RandomElement(0 \mathrel{..} xtzMap[pick.user])$
            )
        $\lor$ LET $matches \triangleq \{x \in orders : x.key[3] = \text{FALSE}\}$
           IN  $\land\ matches \neq \{\}$
               $\land\ ExecuteSellingOrder(RandomElement(matches),$
                      $pick.user,$
                      $RandomElement(0 \mathrel{..} xtzMap[pick.user])$
                      )

$\lor\ \land\ tokenMap[pick.token][pick.user] > 0$
    $\land\ \lor\ CreateSellingOrder(pick.token,$
            $pick.user,$
            $pick.price,$
            $RandomElement(0 \mathrel{..} tokenMap[pick.token][pick.user])$
            )
        $\lor$ LET $matches \triangleq \{x \in orders : x.key[3] = \text{TRUE}\}$
           IN  $\land\ matches \neq \{\}$
               $\land\ ExecuteBuyingOrder(RandomElement(matches),$
                      $pick.user,$
                      $RandomElement(0 \mathrel{..} tokenMap[pick.token][pick.user])$
                      )

$\lor$ UNCHANGED $\langle xtzMap, tokenMap, orders\rangle$
```