2.1 Pizza or Pasta

Module ALICE:

```
Init:
  $custPizzaBuffer = \emptyset$
  $custPastaBuffer = \emptyset$
  $pizzaBuffer = \emptyset$
  $pastaBuffer = \emptyset$

upon ⟨Bob, getJob | NULL⟩
  while ($pizzaBuffer == \emptyset$) {}
  $dish \leftarrow pizzaBuffer$
  trigger ⟨Bob, prepareDish | dish⟩

upon ⟨Carole, getJob | NULL⟩
  while ($pastaBuffer == \emptyset$) {}
  $dish \leftarrow pastaBuffer$
  trigger ⟨Carole, prepareDish | dish⟩

upon ⟨Customer, orderDish | customer, menu⟩ do
  if (menu.sortOf(Pizza))
    $custPizzaBuffer = custPizzaBuffer \cup customer$
    $pizzaBuffer = pizzaBuffer \cup menu$
  else
    $custPastaBuffer = custPastaBuffer \cup customer$
    $pastaBuffer = pastaBuffer \cup menu$

upon ⟨Bob, returnDish | menu⟩ do
  $customer \leftarrow custPizzaBuffer$
  $custPizzaBuffer = custPizzaBuffer \setminus customer$
  trigger ⟨Alice, serveDish | customer, menu⟩

upon ⟨Carole, returnDish | menu⟩ do
  $customer \leftarrow custPastaBuffer$
  $custPastaBuffer = custPastaBuffer \setminus customer$
  trigger ⟨Alice, serveDish | customer, menu⟩
```

Module BOB:

```
Init:
  $bobPizzaBuffer = \emptyset$
  $triggerCounter = 0$

while (TRUE)
  if $|\ bobPizzaBuffer\ | + triggerCounter < 3$
    trigger ⟨Bob, getJob | NULL⟩
    $triggerCounter{+}{+}$
  if ($|\ bobPizzaBuffer\ | \neq 0$)
    $dish \leftarrow bobPizzaBuffer$
    prepare the dish
    $bobPizzaBuffer = bobPizzaBuffer \setminus dish$
    trigger ⟨Bob, returnDish | dish⟩

upon ⟨Bob, prepareDish | dish⟩
  $bobPizzaBuffer = bobPizzaBuffer \cup dish$
  $triggerCounter -= 1$
```

Module CAROLE:

```
Init:
  $carolePastaBuffer = \emptyset$
  $triggerCounter = 0$

while (TRUE)
  if $|\ carolePastaBuffer\ | + triggerCounter < 7$
    trigger ⟨Carole, getJob | NULL⟩
    $triggerCounter{+}{+}$
  if ($|\ carolePastaBuffer\ | \neq 0$)
    $dish \leftarrow carolePastaBuffer$
    prepare the dish
    $carolePastaBuffer = carolePastaBuffer \setminus dish$
    trigger ⟨Carole, returnDish | dish⟩

upon ⟨Carole, prepareDish | dish⟩
  $carolePastaBuffer = carolePastaBuffer \cup dish$
  $triggerCounter -= 1$
```

2.2 Safety and liveness

(a) *If some general attacks at time t, then the other general attacks at the same time.*
This is a ***safety property***, because

(b) *If m_2 arrives after time t, then General A attack after General B.*
This is a ***safety property***, because this statement ensures that General A will attack if and only if it receives the message delivered by m_2 and not just out of thin air.

(c) *Eventually, General B will attack.*
This is a ***liveness property***, because

(d) *If messenegers m_1 and m_2 are not intercepted, then eventually both generals attack.*
This is a ***liveness property***, because

(e) *If m_1 and m_2 are not intercepted, then eventually both generals attack at time t.*
This is a ***mixture***, because we have the safety property that both messenger will arrive in time - so before time t - and the liveness property that eventually both attack at the same time t.

2.3 Unreliable clocks

(a) *Find two examples, where timing issues lead to safety violations.*

(b) *Find two examples, where timing issues lead to liveness volations.*
As a first example, we can take the case from exercise 2.2. In there it could happen that m_2 is delivering the message after the time t. Therefore General A will attack after General B which violates the liveness property.