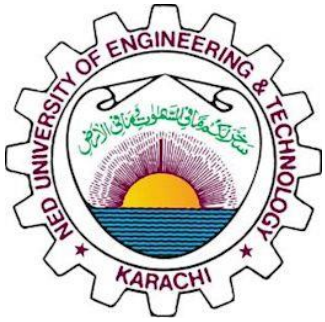


OPERATING SYSTEMS (CS-329)

COMPLEX ENGINEERING PROBLEM REPORT

GROUP MEMBERS

MAHNOOR ZIA – CS22101
ANOOSHA KHALID – CS22104
LAIBA IQRAR – CS22112
SUBMITTED TO: DR. UROOJ AINUDDIN

MAY 4, 2025
NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

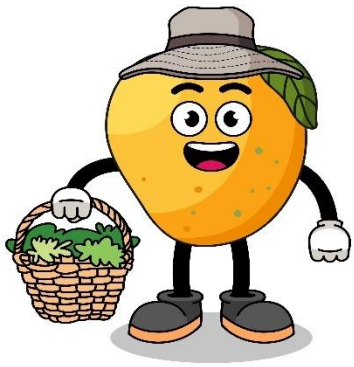
PROBLEM STATEMENT

SPRING WORKERS

Simulate a tree laden with fruits. Launch three “picker” processes and a “loader” process in parallel. If there is fruit on the tree, the picker picks it and places it in a slot of a crate with 12 slots. When a picker finds that the crate is full, it calls the loader. It waits for the loader to place this crate in a truck. Then, the loader furnishes a new crate for the pickers. We assume there is enough space in the truck for all crates. All pickers return to the main function when the tree is bare. In the end, the loader places any partially filled crate in the truck if present. If a picker is adding to the last crate, the loader waits for it to complete the action.

Points to note:

- The number of fruits on the tree is known globally.
- This tree is implemented as an integer array to represent different pieces of fruit.
- The main function provides a shared empty crate when execution starts.
- A piece of fruit can be picked only once and by only one picker for obvious reasons.



CODE OF SPRING WORKERS PROBLEM

spring-workers.py

```python
# Group Members: B'22
# Mahnoor Zia (CS-22101)
# Anoosha Khalid (CS-22104)
# Laiba Iqrar (CS-22112)


import threading
import time
import random
from datetime import datetime



# GLOBAL VARIABLES
CRATE_CAPACITY = 12
TOTAL_FRUITS = 60



# COLORS FOR PRINTING READABILITY
COLOR_PINK = "\033[38;5;213m"
COLOR_BRIGHT_WHITE = "\033[97m"
COLOR_GREEN = "\033[92m"
COLOR_BLUE = "\033[94m"
COLOR_CYAN = "\033[96m"
COLOR_YELLOW = "\033[93m"
COLOR_RESET = "\033[0m"

```

```python
# LOGGER FUNCTION FOR TRACKING
def log(message, section="", indent=0):
    timestamp = datetime.now().strftime("%H:%M:%S")
    section_labels = {
        "picker": f"\n{COLOR_BLUE}[ PICKER ACTIVITY ]{COLOR_RESET}",
        "loader": f"\n{COLOR_GREEN}[ LOADER ACTIVITY ]{COLOR_RESET}",
        "tree": f"\n{COLOR_CYAN}[ FRUIT TREE ]{COLOR_RESET}",
        "final": f"\n{COLOR_YELLOW}[ FINAL SUMMARY ]{COLOR_RESET}"
    }
    label = section_labels.get(section, "")

    if label:
        print(label)
    print(f"{' ' * indent}{COLOR_YELLOW}[{timestamp}]{COLOR_RESET} {message}")



#SEMAPHORES AND MUTEXES
mutex = threading.Lock()                    # for mutual exclusion
semaphore_loader = threading.Semaphore(0)   # Loader waits on this until crate is full
semaphore_picker = threading.Semaphore(0)   # Pickers wait for a new crate after the loader takes the full one.



# SHARED RESOURCES
tree = list(range(1, TOTAL_FRUITS + 1))     #array
crate = []
truck = []
pickers = 3
pickers_in_critical_section = 0
```

```python
#PICKER THREAD
def picker(picker_id):
    global pickers, pickers_in_critical_section
    picker_names = {1: "Anoosha", 2: "Laiba", 3: "Mahnoor"}
    picker_name = picker_names[picker_id]

    while True:
        mutex.acquire()  # semWait(mutex)

        pickers_in_critical_section += 1

        if not tree:  # No more fruits left
            pickers_in_critical_section -= 1

            pickers -= 1
            if TOTAL_FRUITS == 0:
                log("OOPS! No fruits available on the tree :( No need to call the loader.", section="tree")
                print(" " * 4 + f"{picker_name} is upset and exiting.")
            else:
                log(f"{picker_name} has finished picking and is waiting for loader to finish.", section="picker", indent=4)
                print(" " * 4 + "Tree is bare.")

            semaphore_loader.release()       # semSignal(L)
            mutex.release()             # semSignal(mutex)
            return


        if len(crate) == CRATE_CAPACITY:
            pickers_in_critical_section -= 1
            mutex.release()             # semSignal(mutex)
            semaphore_picker.acquire()  # semWait(P)
            continue


        # Pick a fruit
        fruit = tree.pop(0)
        crate.append(fruit)
        log(f"{picker_name} picked fruit {fruit}.", section="picker", indent=4)
        print(" " * 4 + f"Current crate size: {len(crate)}/{CRATE_CAPACITY}")


        if len(crate) == CRATE_CAPACITY:        # Notify loader once crate is full
            log(f"{picker_name} has filled the crate with {CRATE_CAPACITY} fruits.", section="picker", indent=4)
            print(" " * 4 + "Found crate full. Notifying loader.")
            semaphore_loader.release()         # semSignal(L)

        pickers_in_critical_section -= 1
        mutex.release()                        # semSignal(mutex)
        time.sleep(random.uniform(0.05, 0.2))  # to alternation of pickers


# LOADER THREAD
def loader():
    while True:
        semaphore_loader.acquire()      # semWait(L)
        mutex.acquire()                 # semWait(mutex)

        # Check if the crate is full
        if len(crate) == CRATE_CAPACITY:
            log("Loader triggered! Crate is full.", section="loader", indent=2)
            print(" " * 4 + "Loading it to truck...")
            truck.append(crate[:])
            crate.clear()

            # It will notify pickers that they can start working on a new crate
            for _ in range(pickers):
                semaphore_picker.release()        # semSignal(P)

            mutex.release()                       # semSignal(mutex)
            continue


        # If all pickers are done and there's a partial crate
        if pickers == 0 and pickers_in_critical_section == 0 and crate:
            log("Loader detected partially filled crate after pickers finished.", section="loader", indent=2)
            print(" " * 4 + "Loader is moving the final partial crate to the truck.")
            truck.append(crate[:])
            crate.clear()
            if TOTAL_FRUITS == 0:
```

```
            if TOTAL_FRUITS == 0:
                return
            else:
                log("Loader has completed all operations and is exiting.", section="loader", indent=2)
            mutex.release()                         # semSignal(mutex)
            return


        # If no pickers left and no crate, finish
        if pickers == 0 and pickers_in_critical_section == 0 and not crate:
            if TOTAL_FRUITS == 0:
                return
            else:
                log("Loader has completed all operations and is exiting.", section="loader", indent=2)
            mutex.release()                         # semSignal(mutex)
            return

        mutex.release()                             # semSignal(mutex)


#MAIN
def main():
    print("\n┌──────────────────────────────────────────────────────┐")
    print(f"          {COLOR_PINK}  SPRING WORKERS SIMULATION START  {COLOR_RESET}")
    print("└──────────────────────────────────────────────────────┘")

    if TOTAL_FRUITS < 0:    # if enter the neg no for fruits
        print("\nOOPS! Fruits can't be negative.\nENTER THE ACCURATE DETAILS PLS.")
        print("Exiting the simulation...")
        return

    print("\nYay! Mango season has started, it's time to pluck the mangoes from the tree!")
    print("Pickers: 1 - Anoosha | 2 - Laiba | 3 - Mahnoor\n")

    for i in range(0, len(tree), 10):       # for printing the tree
        print(" " * 4 + ' '.join(map(str, tree[i:i + 10])))


    #creating threads
    picker_threads = [threading.Thread(target=picker, args=(i,)) for i in range(1, 4)]
    loader_thread = threading.Thread(target=loader)


    # start the both picker and loader to finish their work
    for t in picker_threads:
        t.start()
    loader_thread.start()


    # wait for both picker and loader to finish their work
    for t in picker_threads:
        t.join()
    loader_thread.join()

    log("", section="final")


    # printing the summary
    if TOTAL_FRUITS > 0:
        print("\nCrates in the Truck:")
    for index, crate in enumerate(truck, 1):
        print(f"\n[ Crate {index} ]")


        print(f"{COLOR_GREEN}┌──────────────────────────────────────────┐{COLOR_RESET}")
        for i in range(0, len(crate), 12):
            fruits_row = ' '.join(map(str, crate[i:i+12]))
            print(f" {fruits_row:<40} ")
        print(f"{COLOR_GREEN}└──────────────────────────────────────────┘{COLOR_RESET}")
        print(f"({len(crate)} fruits)")


    print(f"\nTotal crates loaded: {len(truck)}")
    print(f"{COLOR_GREEN}Spring harvest has been successfully completed. Thank you, workers!{COLOR_RESET}\n")

main()
```

TEST CASES

CASE 1:

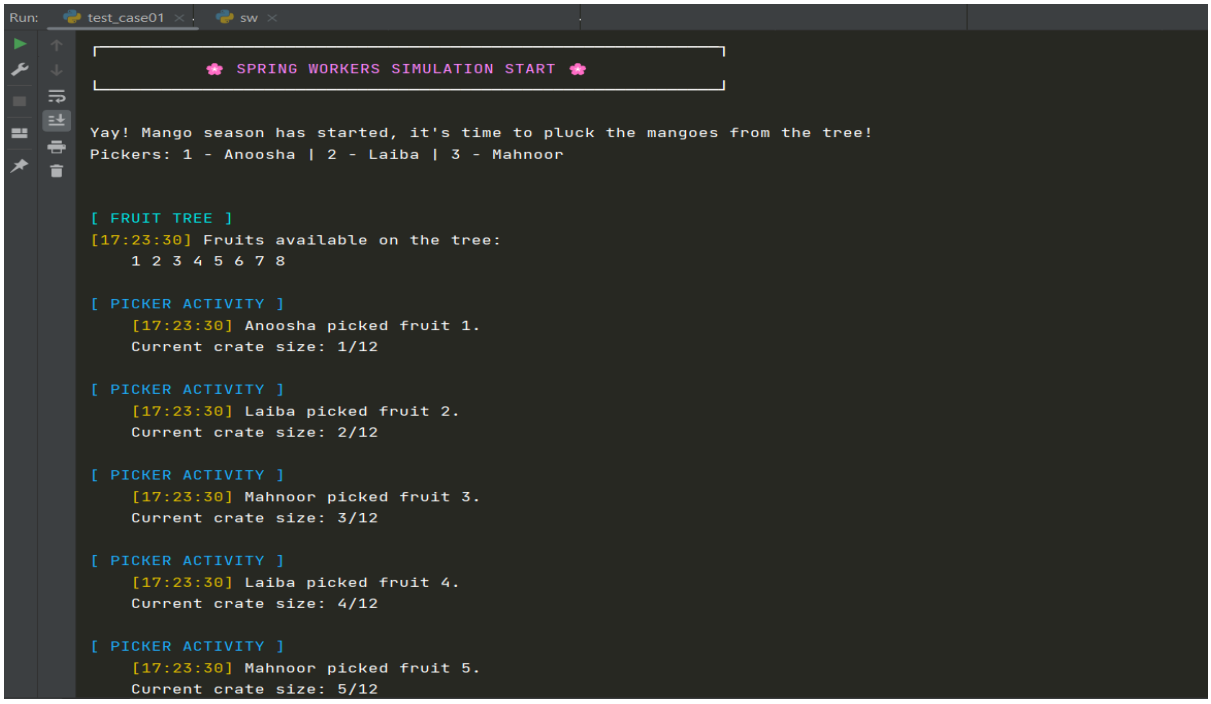
[Checking the partial crate logic]

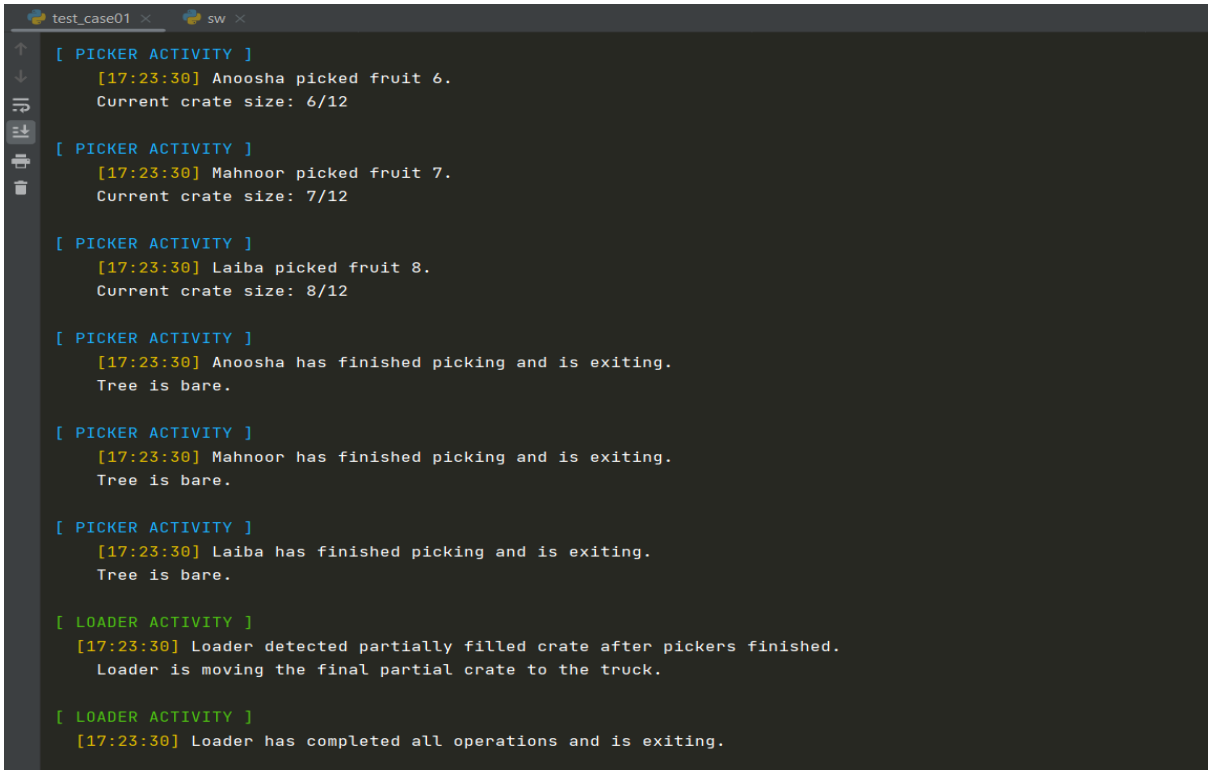
- TOTAL_FRUITS = 8
- CRATE_CAPACITY = 12
- pickers = 3

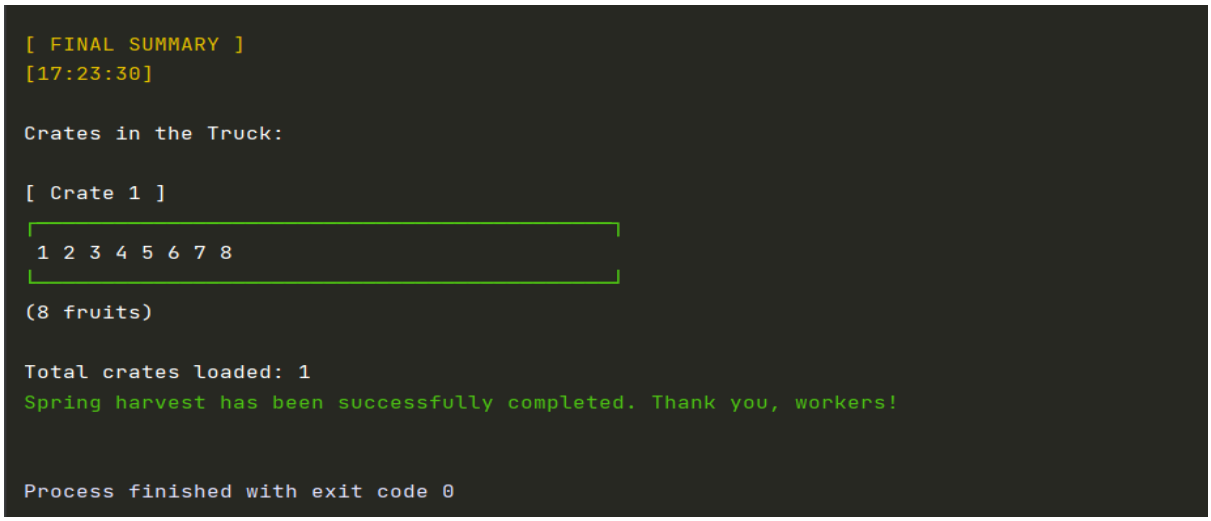
Expected output: Since there are fewer than 12 fruits, the crate will be partially filled with 8 fruits. After all pickers finish, the loader will move the partial crate to the truck, completing the task as expected.

STATUS PASSED!

Actual Output:







CASE 2:

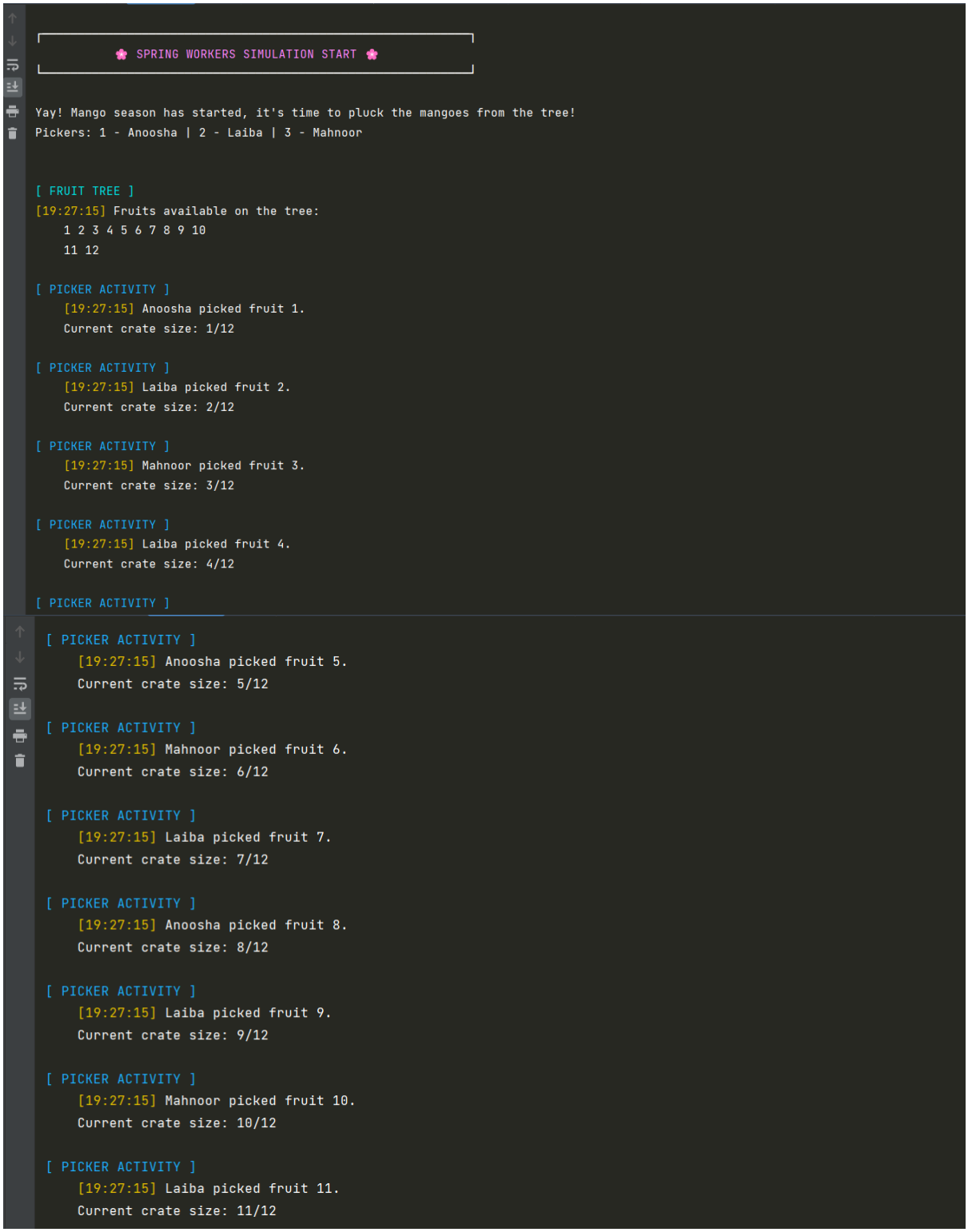
[Checking the full crate logic]

- TOTAL_FRUITS = 12
- CRATE_CAPACITY = 12
- pickers = 3

Expected output: Pickers (3) fill crates in parallel, and each crate is moved to the truck once it reaches 12 fruits. No partial crate will be detected.

STATUS PASSED!

Actual Output:



```
[ PICKER ACTIVITY ]
    [19:27:15] Mahnoor picked fruit 12.
    Current crate size: 12/12

[ PICKER ACTIVITY ]
    [19:27:15] Mahnoor has filled the crate with 12 fruits.
    Found crate full. Notifying loader.

[ LOADER ACTIVITY ]
  [19:27:15] Loader triggered! Crate is full.
    Loading it to truck...

[ PICKER ACTIVITY ]
    [19:27:15] Anoosha has finished picking and is waiting for loader to finish.
    Tree is bare.

[ PICKER ACTIVITY ]
    [19:27:15] Laiba has finished picking and is waiting for loader to finish.
    Tree is bare.

[ PICKER ACTIVITY ]
    [19:27:15] Mahnoor has finished picking and is waiting for loader to finish.
    Tree is bare.

[ LOADER ACTIVITY ]
  [19:27:16] Loader has completed all operations and is exiting.

  [ LOADER ACTIVITY ]
    [19:27:16] Loader has completed all operations and is exiting.

  [ FINAL SUMMARY ]
  [19:27:16]

  Crates in the Truck:

  [ Crate 1 ]
  ┌──────────────────────────────────────────┐
   1 2 3 4 5 6 7 8 9 10 11 12
  └──────────────────────────────────────────┘
  (12 fruits)

  Total crates loaded: 1
  Spring harvest has been successfully completed. Thank you, workers!


  Process finished with exit code 0
```

CASE 3:

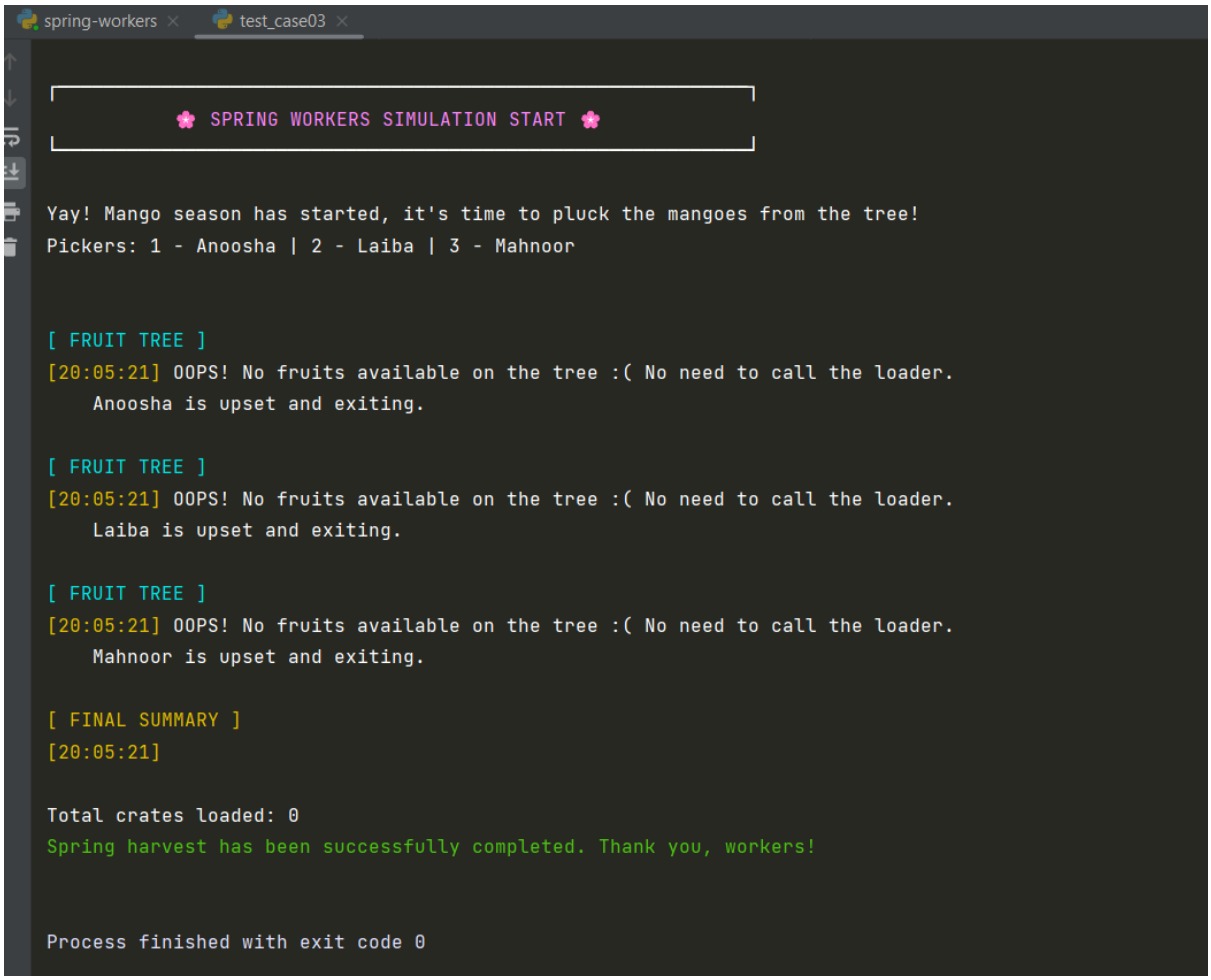
[if there's no fruit on the tree]

- TOTAL_FRUITS = 0
- CRATE_CAPACITY = 12
- pickers = 3

Expected output: Pickers will immediately finish and will not call the loader since there are no fruits.

STATUS PASSED!

Actual Output:



CASE 4:
[if fruit count is in negative number]

- TOTAL_FRUITS = -5
- CRATE_CAPACITY = 12
- pickers = 3

Expected output: Immediate exit.

STATUS PASSED!

Actual Output: