STAGE 8 – WORKER ALIENS

The worker aliens are placed in random locations at the beginning of the game. In the lore, it is suggested that alien workers are produced by the queen asexually. While it would be possible to program this, including movement from the queen's location, it is not necessary. Often, providing NPC actions are believable, they don't need to follow the rules absolutely.

For example, in racing games, NPC cars in front will often slow down, allowing the player to catch up – this makes the game more enjoyable, and as long as the cars don't stop completely, the movement looks believable. In this game, since the player cannot know what is happening elsewhere in the space station, it is entirely believable that the queen is spawning workers and they are moving.

The worker aliens provide a second mechanism by which the player can die. They either run out of moves – i.e., the space station power is exhausted – or are killed by a worker alien, adding challenge to the game.

When a worker alien is encountered, the player has two choices – scare the alien away or kill it. Frightening a worker enables the player to continue, but the creature remains in the module for a future encounter. Killing it removes it from the game. Both actions require the player to use the flamethrower.

The player decides how much fuel to use. 30 to 80 units are required to frighten a worker alien, while 90 to 140 units are required to kill it. However, the player won't know this and will need to learn through experience. The random element can cause the player to panic and use either too little or too much fuel.

If a player runs out of fuel during an encounter, the alien will attack and kill them. If the player uses too little fuel for their chosen action, they must try again. It would be possible to build a hit-point mechanism into the game if able students wanted to do that.

In the base game, the player cannot escape an encounter until they either frighten the worker alien or kill it.

Try TIME

Add the following procedure to your code:

```
def worker_aliens():
    global module, workers, fuel, alive
    #Output alien encountered
    if module in workers:
        print("Startled, a young alien scuttles across the floor.")
        print("It turns and leaps towards us.")
        #Get the player's action
        successful_attack = False
        while successful_attack == False:
            print("You can:")
            print()
            print("- Short blast your flamethrower to frighten it away.")
            print("- Long blast your flamethrower to try to kill it.")
            print()
            print("How will you react? (S, L)")
            action = 0
            while action not in ("S", "L"):
                action = input("Press the trigger: ")
            fuel_used = int(input("How much fuel will you use? ..."))
            fuel = fuel - fuel_used
            #Check if player has run out of fuel
            if fuel <= 0:
                alive = False
                return
            #Work out how much fuel is needed
            if action == "S":
                fuel_needed = 30 + 10*random.randint(0,5)
            if action == "L":
                fuel_needed = 90 + 10*random.randint(0,5)
            #Try again if not enough fuel was used
            if fuel_used >= fuel_needed:
                successful_attack = True
            else:
                print("The alien squeals but is not dead. It's angry.")
        #Successful action
        if action == "S":
            print("The alien scuttles away into the corner of the room.")
        if action == "L":
            print("The alien has been destroyed.")
            #Remove the worker from the module
            workers.remove(module)
        print()
```

0 tabs | 1 tab | 2 tabs | 3 tabs | 4 tabs

In the main program section, underneath `move_queen()`, add:

```
worker_aliens()
```

Investigate TIME

ITEM 	Q: Identify the name of one variable from the `worker_aliens()` subroutine that is storing a Boolean data type.

STRUCTURE 	Q: Here is a line of code from the `worker_aliens()` subroutine:
	`if fuel <= 0:`
	Why could we not have used the following line instead?
	`if fuel < 0:`

PURPOSE	Q: What is the purpose of the following `while` loop?
	`while action not in ("S", "L"):`
	`  action = input("Press the trigger: ")`

REASON 	Q: Why has a double equals sign (==) been used in the following line of code instead of a single equals sign (=)? `if action == "S":`
RELATION 	Q: We placed the call to the `worker_aliens()` subroutine underneath the `move_queen()` line of code in the main program section. Did it have to be placed here? Could it be moved somewhere else?
APPROACH 	Q: How would go about modifying the `worker_aliens()` subroutine so that when an alien is frightened away but not killed, it can escape to another random module that does not contain another NPC?

Make TIME

1 point Add a comment above the `worker_aliens()` subroutine that summarises the purpose of the new module.
2 points Add validation to prevent the program from crashing if the user enters a string when prompted for how much fuel to use.
2 points Play the game as a real player to check the balance. Modify the variables as required to achieve a fun but challenging game.
3 points Add the option to RUN when a worker alien is encountered. If a player chooses to do this, they return to the last module unless they arrived via a ventilation shaft – in which case, they retreat to a random connected module. There is a small possibility that the player will be attacked and die if they do this.
3 points In the game, there is a small group of four astronauts aboard the space station. If fuel runs out during an encounter or the player retreats but is killed, the game ends. If there are characters remaining when one player dies, they escape to a random connected module and get a new flamethrower with 250 fuel. When the last astronaut dies, or the space station power reaches zero, the game is over.

Evaluate TIME

1. A game like this will need balancing. What does that mean in the context of computer games? What needs to be considered when balancing this game?