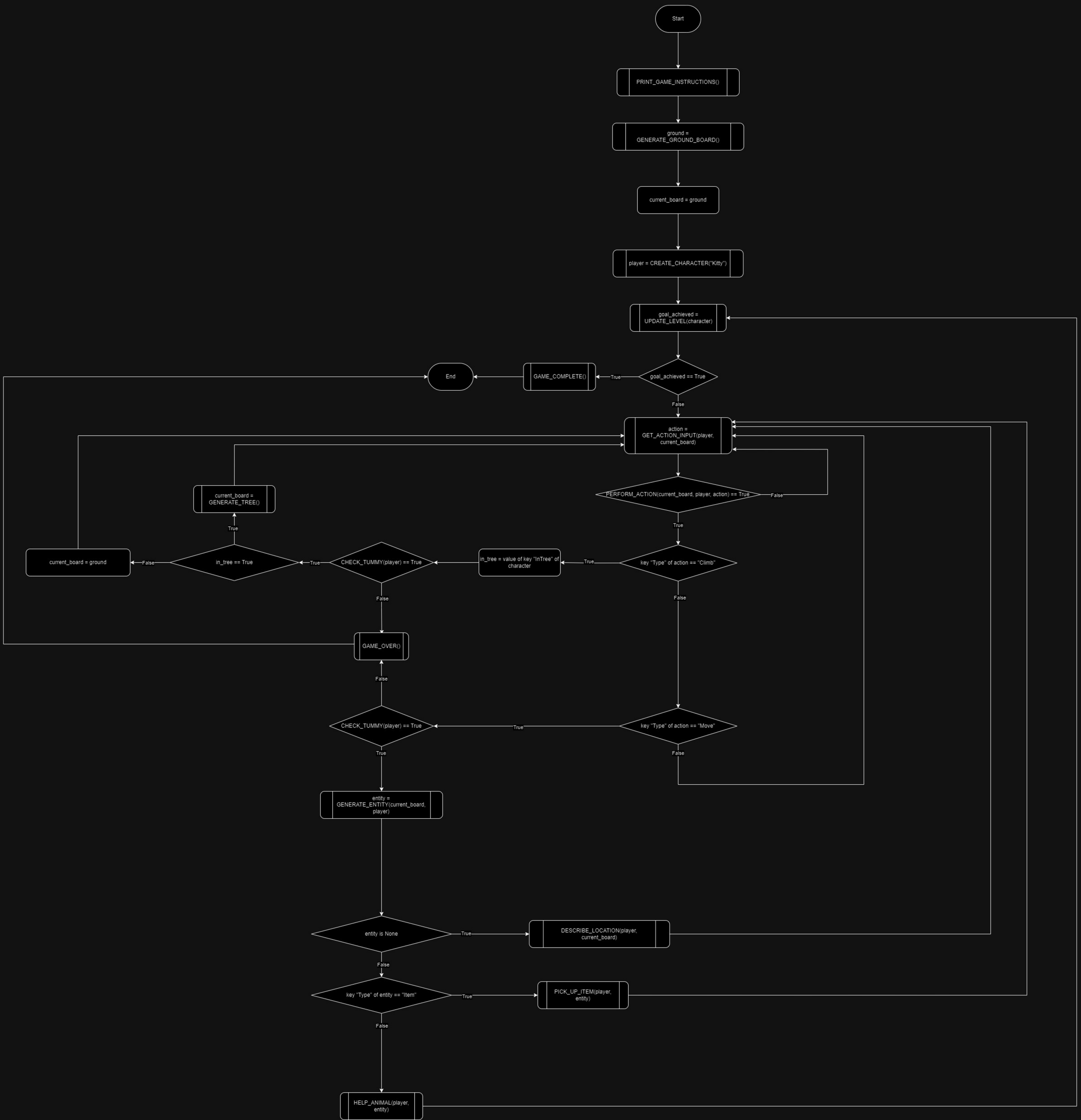
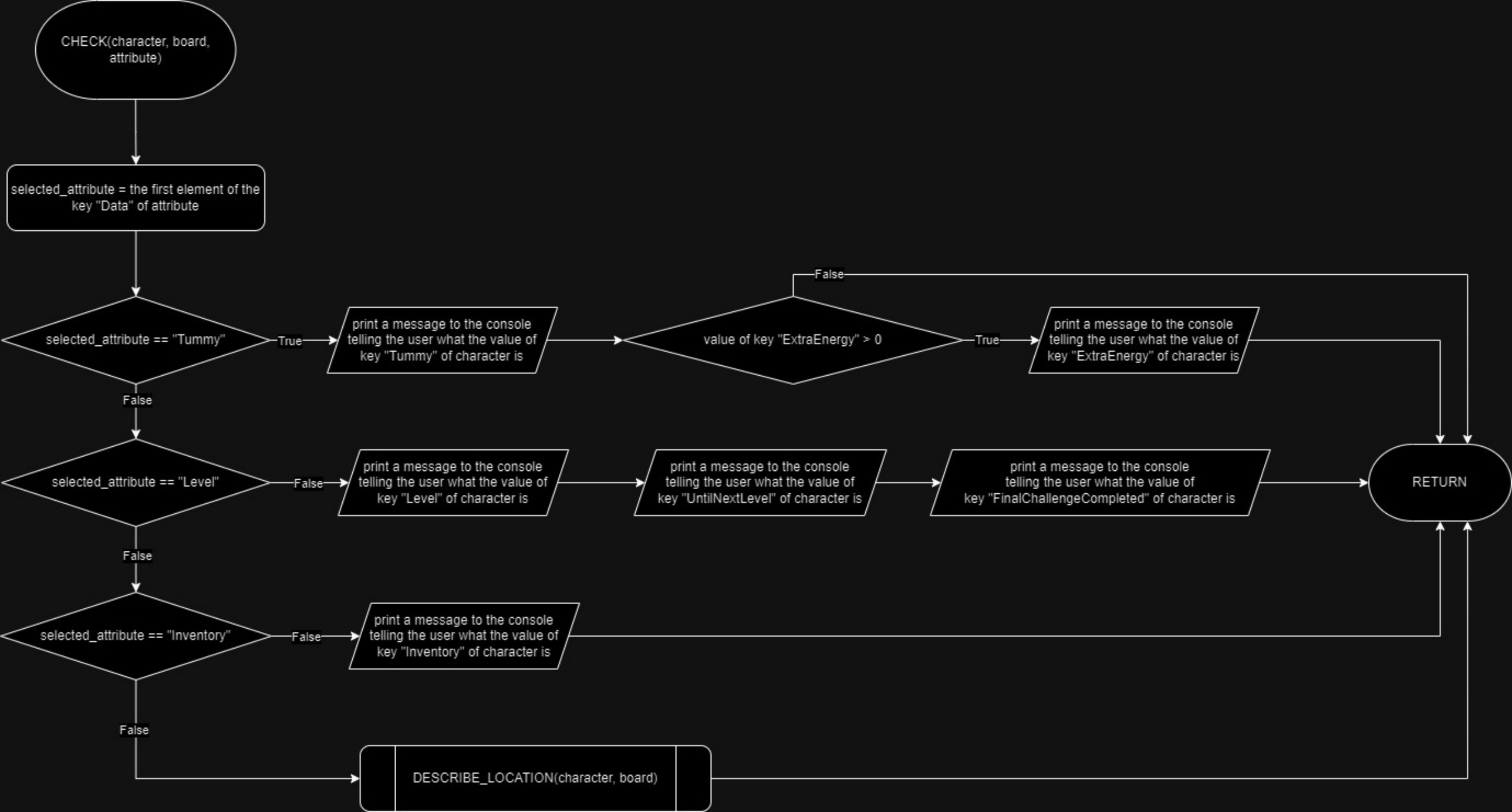
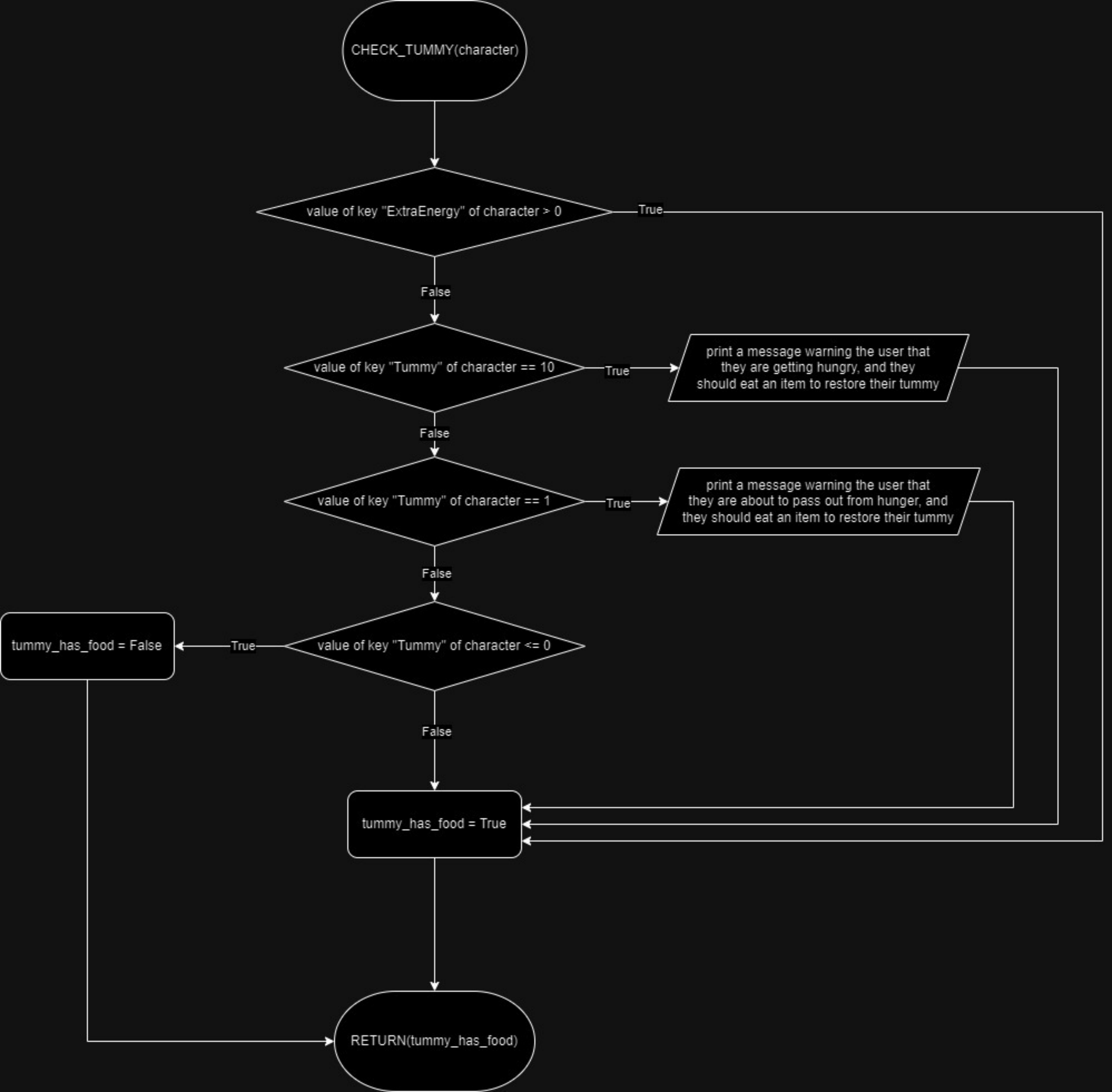
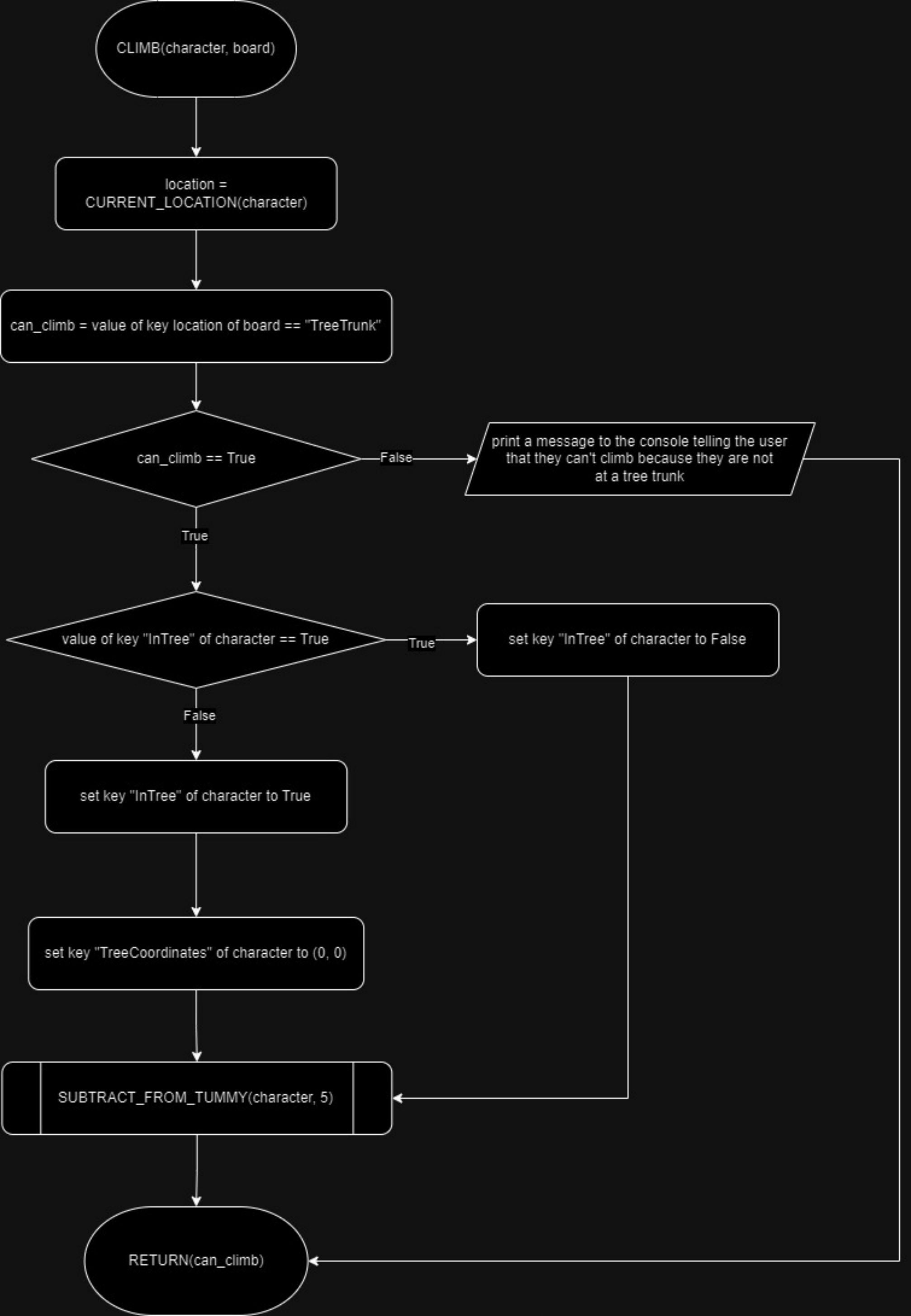
Main Game Function

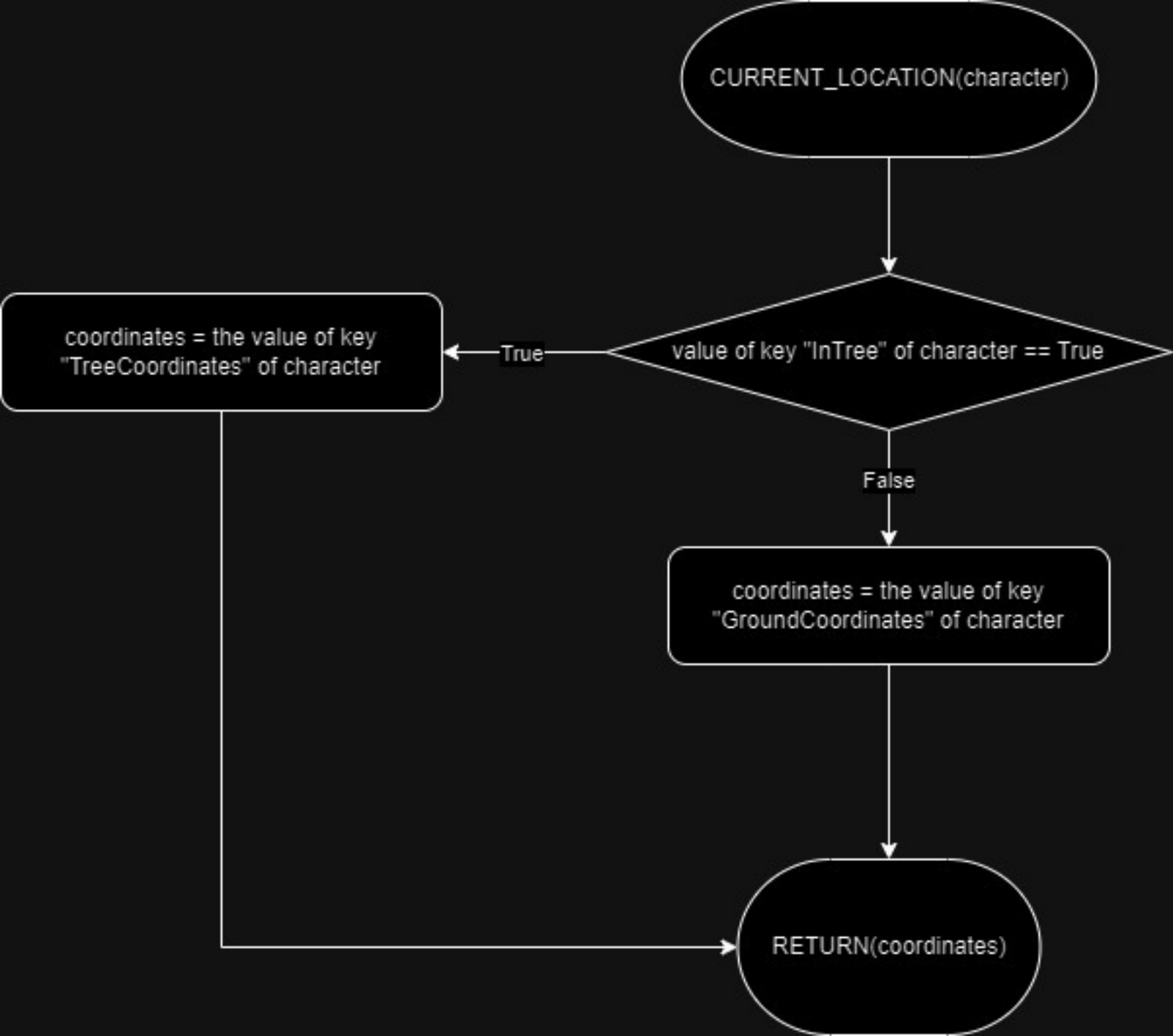


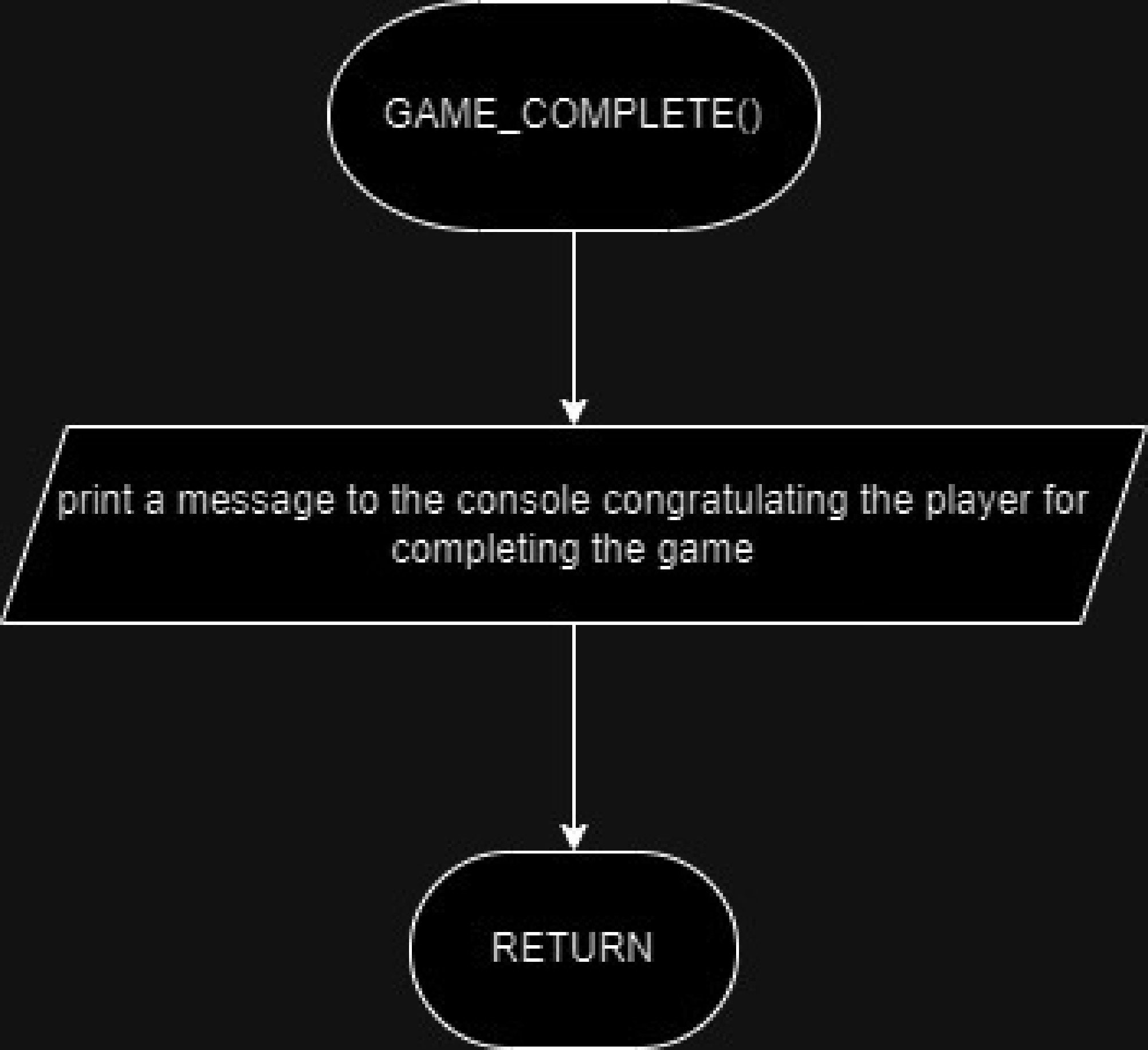

CHECK(character, board, attribute)
selected_attribute = the first element of the key "Data" of attribute
selected_attribute == "Tummy"
True
print a message to the console telling the user what the value of key "Tummy" of character is
value of key "ExtraEnergy" > 0
False
True
print a message to the console telling the user what the value of key "ExtraEnergy" of character is
False
selected_attribute == "Level"
False
print a message to the console telling the user what the value of key "Level" of character is
print a message to the console telling the user what the value of key "UntilNextLevel" of character is
print a message to the console telling the user what the value of key "FinalChallengeCompleted" of character is
RETURN
False
selected_attribute == "Inventory"
False
print a message to the console telling the user what the value of key "Inventory" of character is
False
DESCRIBE_LOCATION(character, board)

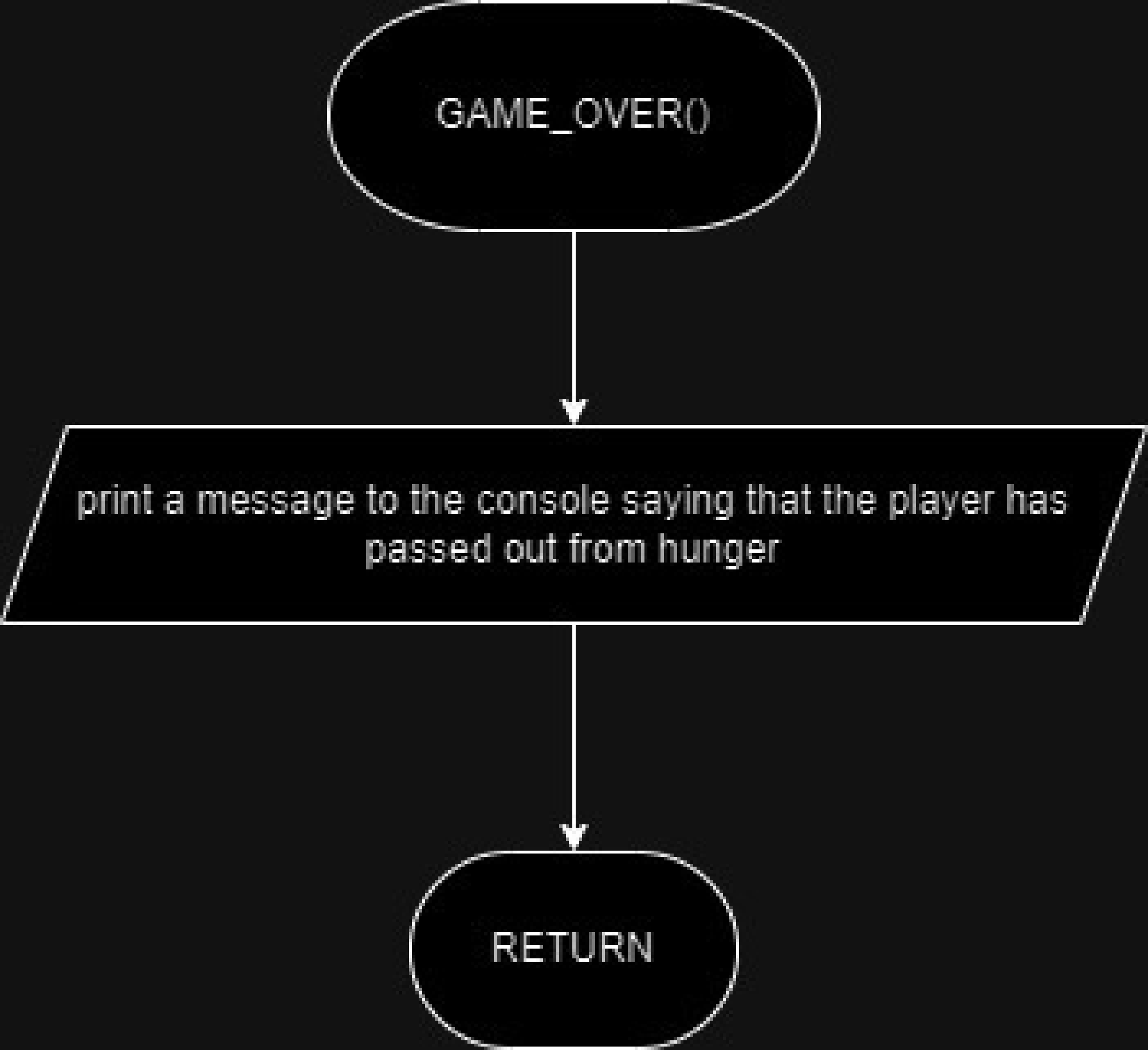

CHECK_TUMMY(character)
value of key "ExtraEnergy" of character > 0
True
False
value of key "Tummy" of character == 10
True
print a message warning the user that they are getting hungry, and they should eat an item to restore their tummy
False
value of key "Tummy" of character == 1
True
print a message warning the user that they are about to pass out from hunger, and they should eat an item to restore their tummy
False
value of key "Tummy" of character <= 0
True
tummy_has_food = False
False
tummy_has_food = True
RETURN(tummy_has_food)

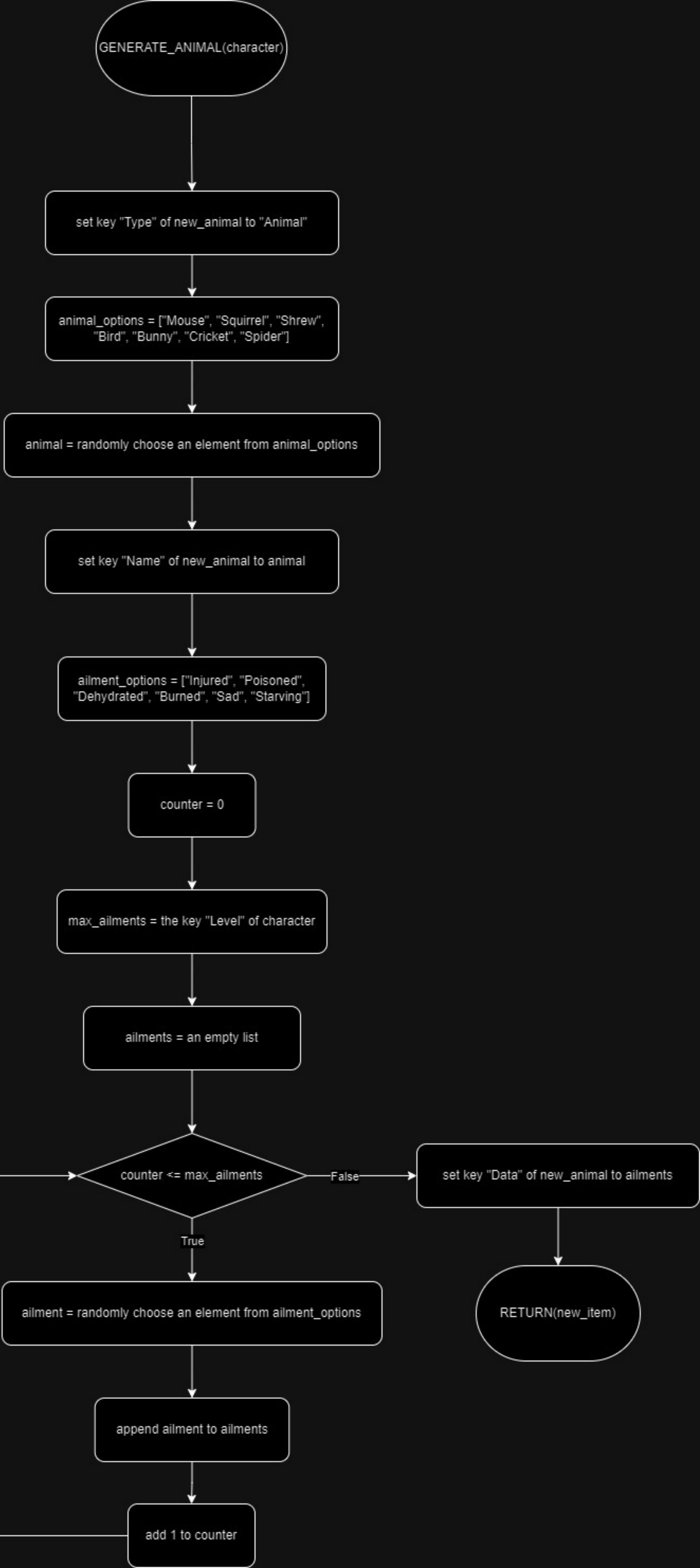

CLIMB(character, board)
location = CURRENT_LOCATION(character)
can_climb = value of key location of board == "TreeTrunk"
can_climb == True
False
print a message to the console telling the user that they can't climb because they are not at a tree trunk
True
value of key "InTree" of character == True
True
set key "InTree" of character to False
False
set key "InTree" of character to True
set key "TreeCoordinates" of character to (0, 0)
SUBTRACT_FROM_TUMMY(character, 5)
RETURN(can_climb)

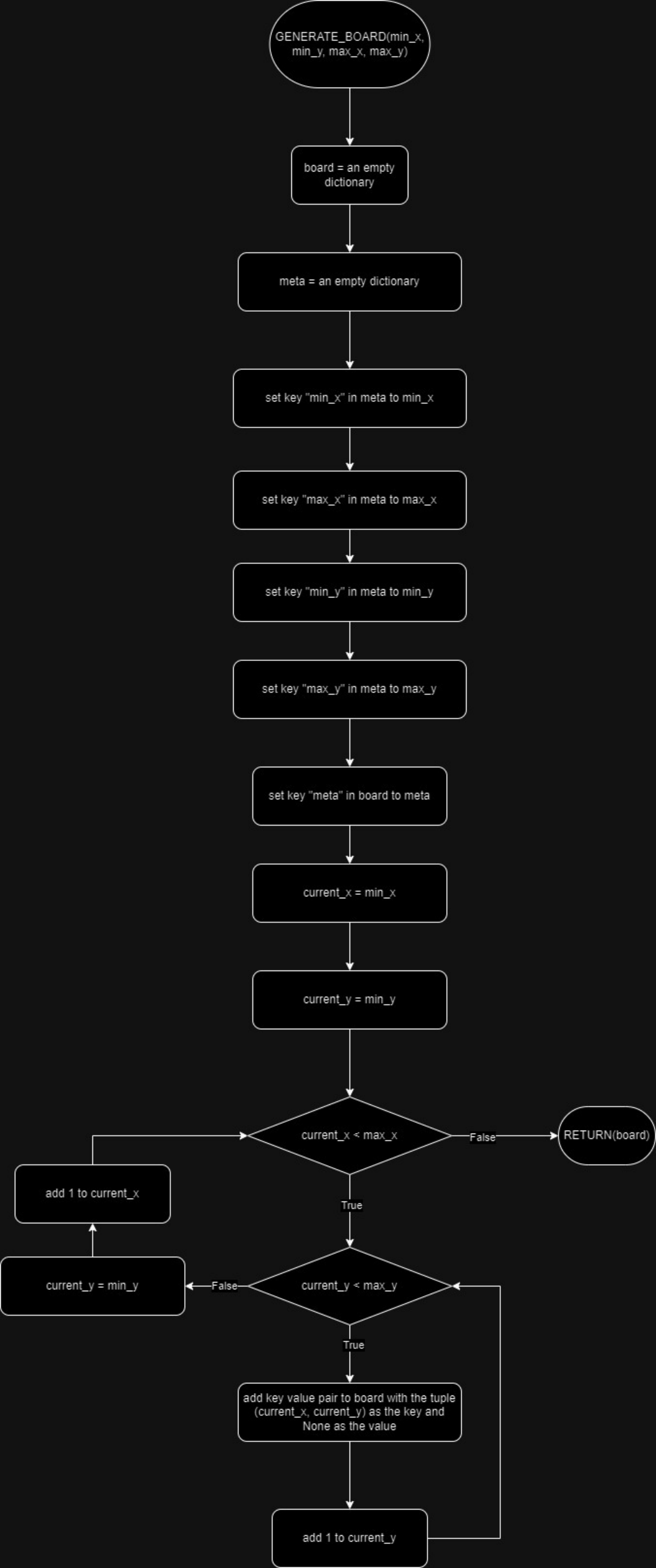
CREATE_CHARACTER(name)
new_character = an empty dictionary
set key "Name" of new_character to name
set key "Level" of new_character to 1
set key "UntilNextLevel" of new_character to 5
set key "InTree" of new_character to False
set key "GroundCoordinates" of new_character to (0, 0)
set key "TreeCoordinates" of new_character to (0, 0)
set key "Tummy" of new_character to 100
set key "AnimalsHelped" to 0
set key "FinalChallengeCompleted" to None
inventory = an empty dictionary
set key "Catnip" of inventory to 0
set key "Silvervine" of inventory to 0
berries = an empty dictionary
set key "Red" of berries to 1
set key "Green" of berries to 1
set key "Blue" of berries to 1
set key "Yellow" of berries to 1
set key "Purple" of berries to 1
set key "Berries" of inventory to berries
set key "Inventory" of new_character to inventory
RETURN(new_character)

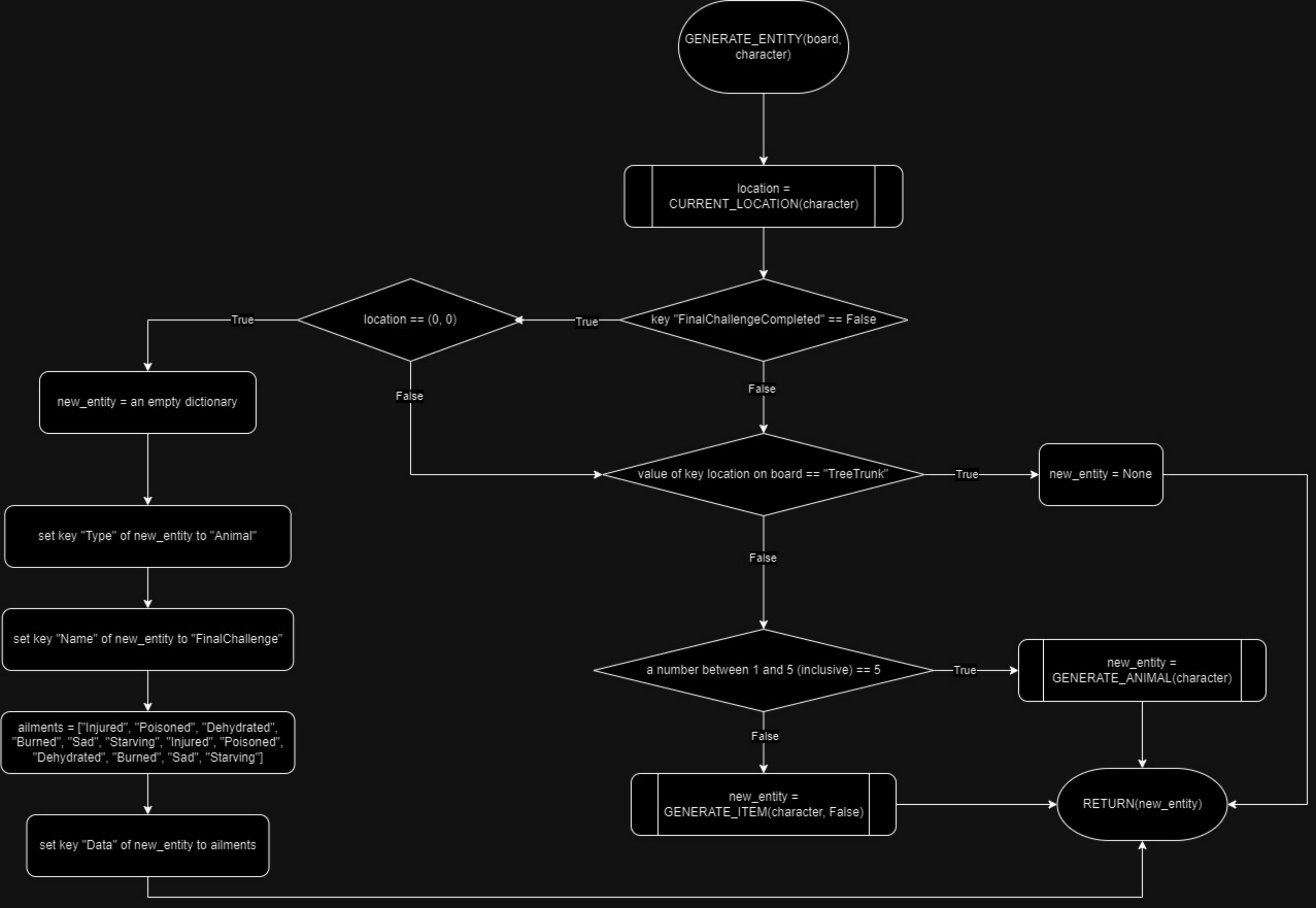

CURRENT_LOCATION(character)
value of key "InTree" of character == True
True
coordinates = the value of key "TreeCoordinates" of character
False
coordinates = the value of key "GroundCoordinates" of character
RETURN(coordinates)

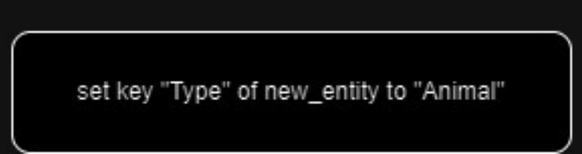

GAME_COMPLETE()
print a message to the console congratulating the player for completing the game
RETURN

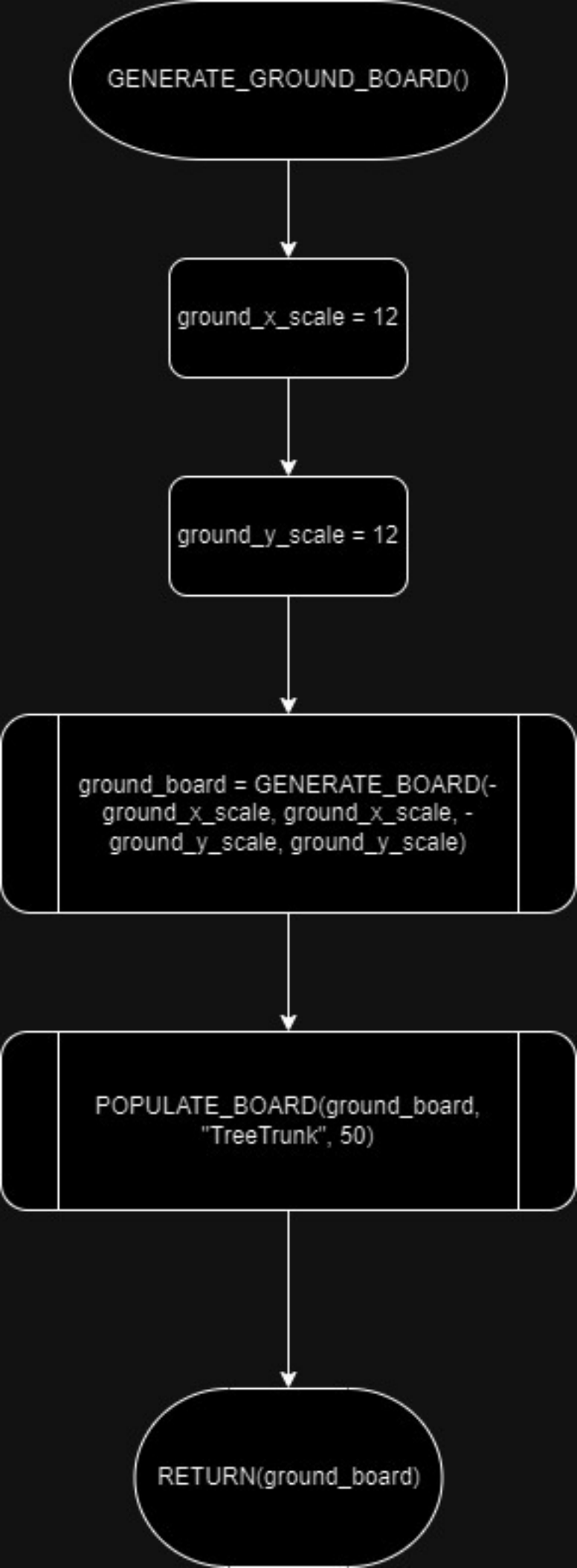

GAME_OVER()
print a message to the console saying that the player has passed out from hunger
RETURN

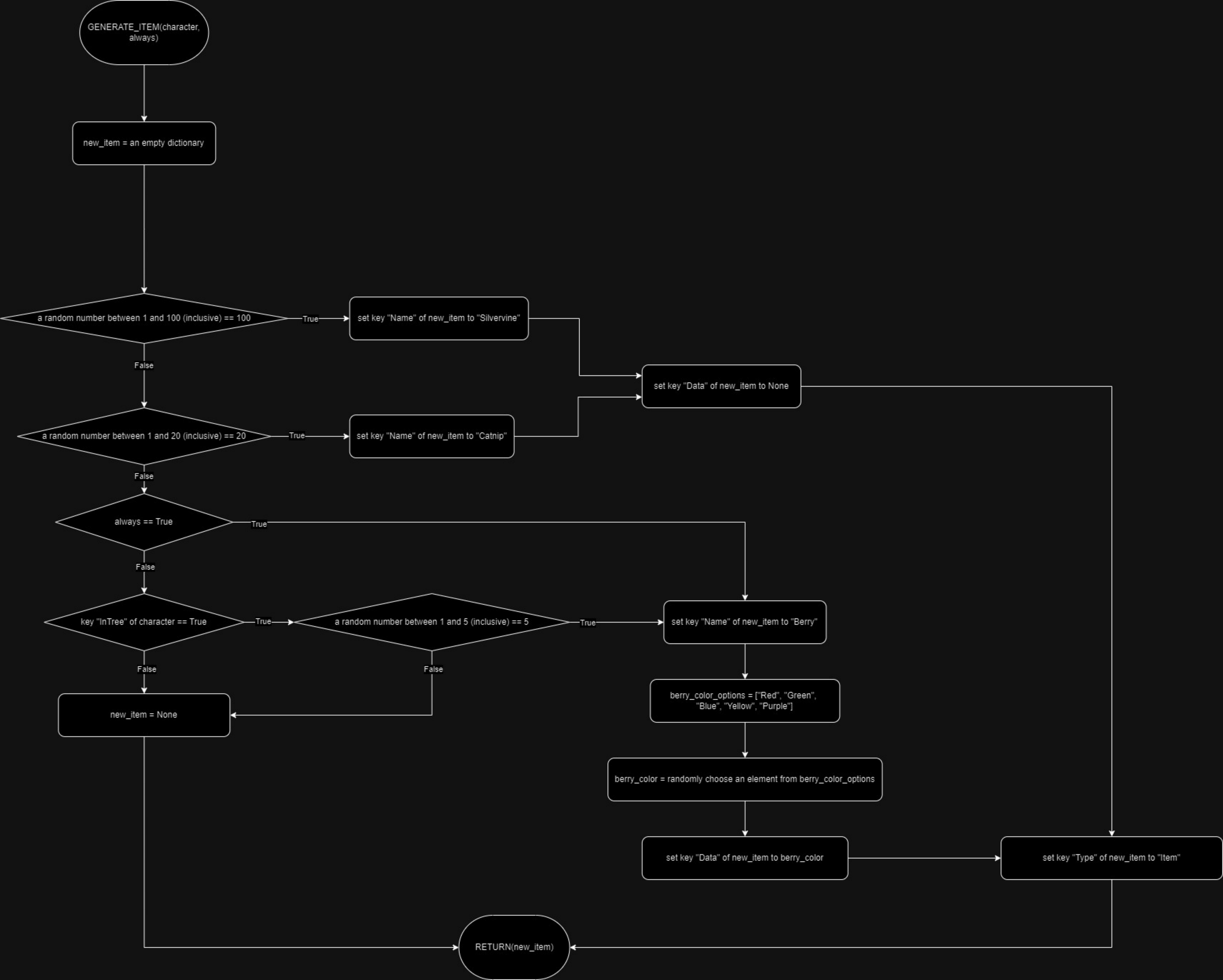

GENERATE_ANIMAL(character)
set key "Type" of new_animal to "Animal"
animal_options = ["Mouse", "Squirrel", "Shrew", "Bird", "Bunny", "Cricket", "Spider"]
animal = randomly choose an element from animal_options
set key "Name" of new_animal to animal
ailment_options = ["Injured", "Poisoned", "Dehydrated", "Burned", "Sad", "Starving"]
counter = 0
max_ailments = the key "Level" of character
ailments = an empty list
counter <= max_ailments
False
set key "Data" of new_animal to ailments
RETURN(new_item)
True
ailment = randomly choose an element from ailment_options
append ailment to ailments
add 1 to counter

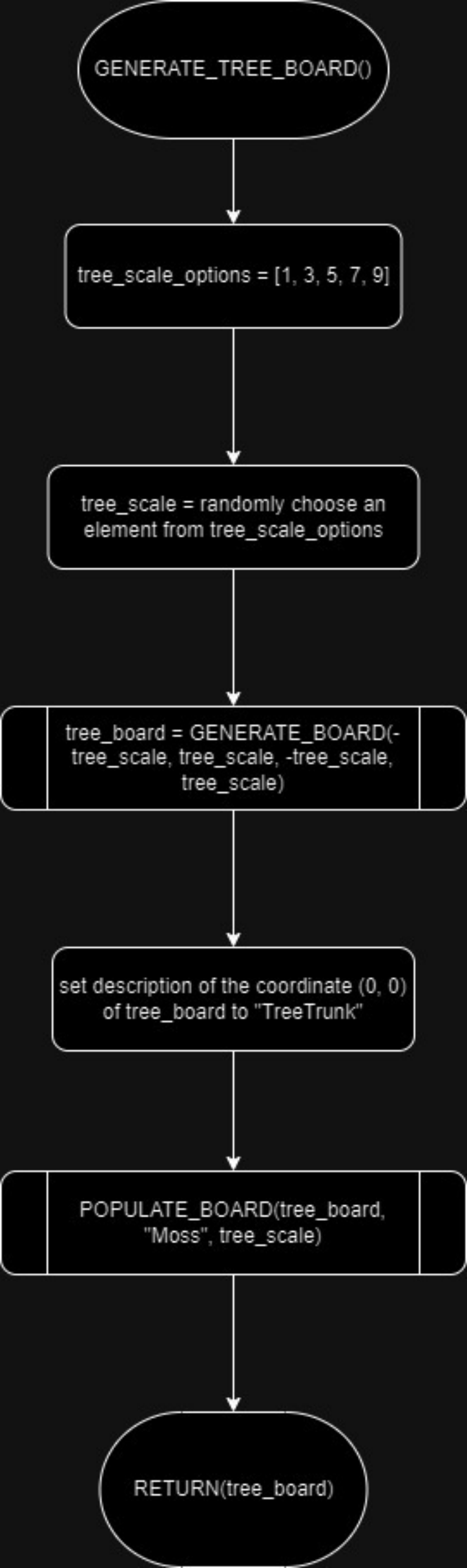

GENERATE_BOARD(min_x, min_y, max_x, max_y)
board = an empty dictionary
meta = an empty dictionary
set key "min_x" in meta to min_x
set key "max_x" in meta to max_x
set key "min_y" in meta to min_y
set key "max_y" in meta to max_y
set key "meta" in board to meta
current_x = min_x
current_y = min_y
current_x < max_x
False
RETURN(board)
True
add 1 to current_x
current_y = min_y
False
current_y < max_y
True
add key value pair to board with the tuple (current_x, current_y) as the key and None as the value
add 1 to current_y

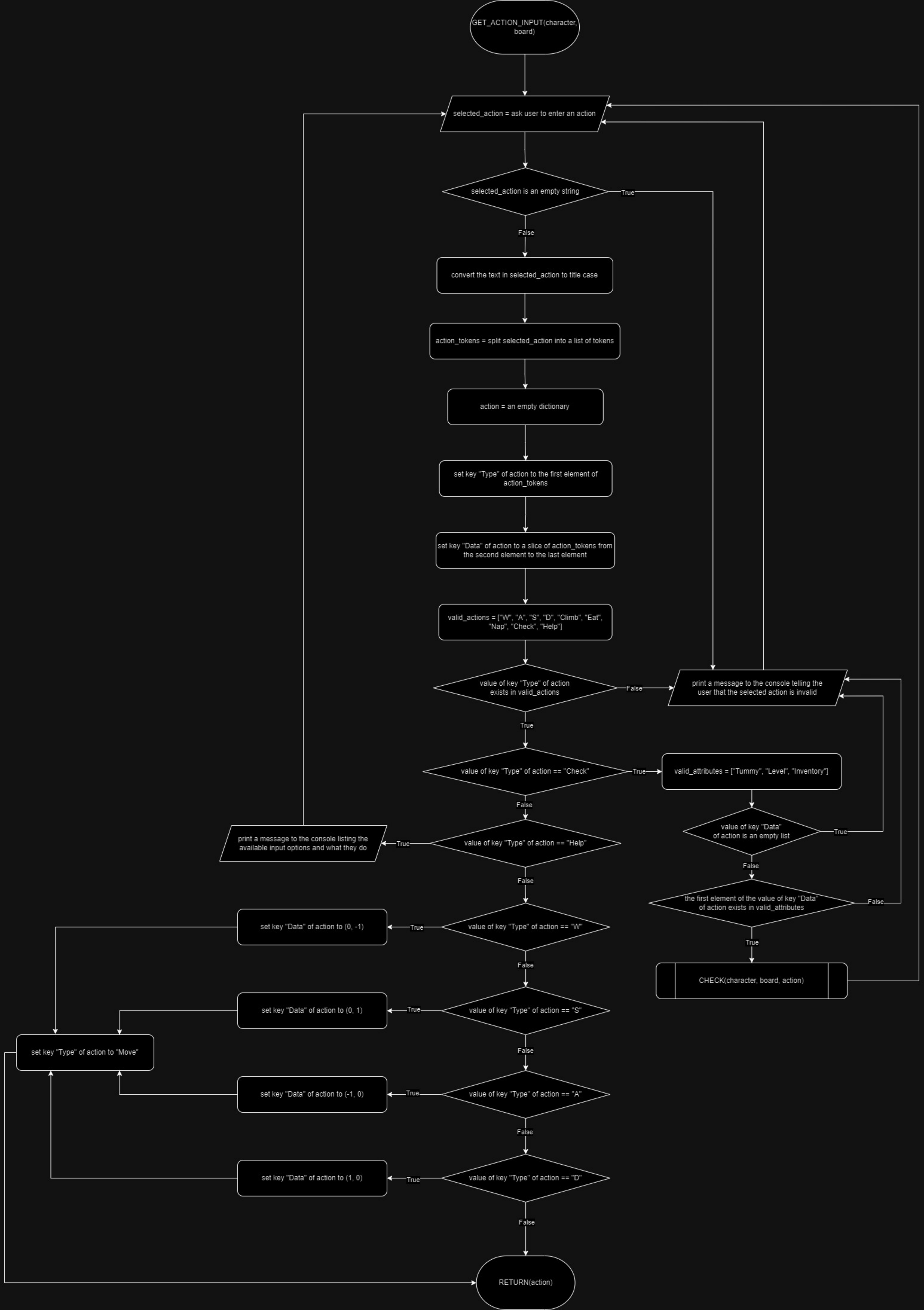

GENERATE_ENTITY(board, character)
location = CURRENT_LOCATION(character)
key "FinalChallengeCompleted" == False
True
location == (0, 0)
True
False
new_entity = an empty dictionary
False
value of key location on board == "TreeTrunk"
True
new_entity = None
set key "Type" of new_entity to "Animal"
False
set key "Name" of new_entity to "FinalChallenge"
a number between 1 and 5 (inclusive) == 5
True
new_entity = GENERATE_ANIMAL(character)
ailments = ["Injured", "Poisoned", "Dehydrated", "Burned", "Sad", "Starving", "Injured", "Poisoned", "Dehydrated", "Burned", "Sad", "Starving"]
False
new_entity = GENERATE_ITEM(character, False)
RETURN(new_entity)
set key "Data" of new_entity to ailments

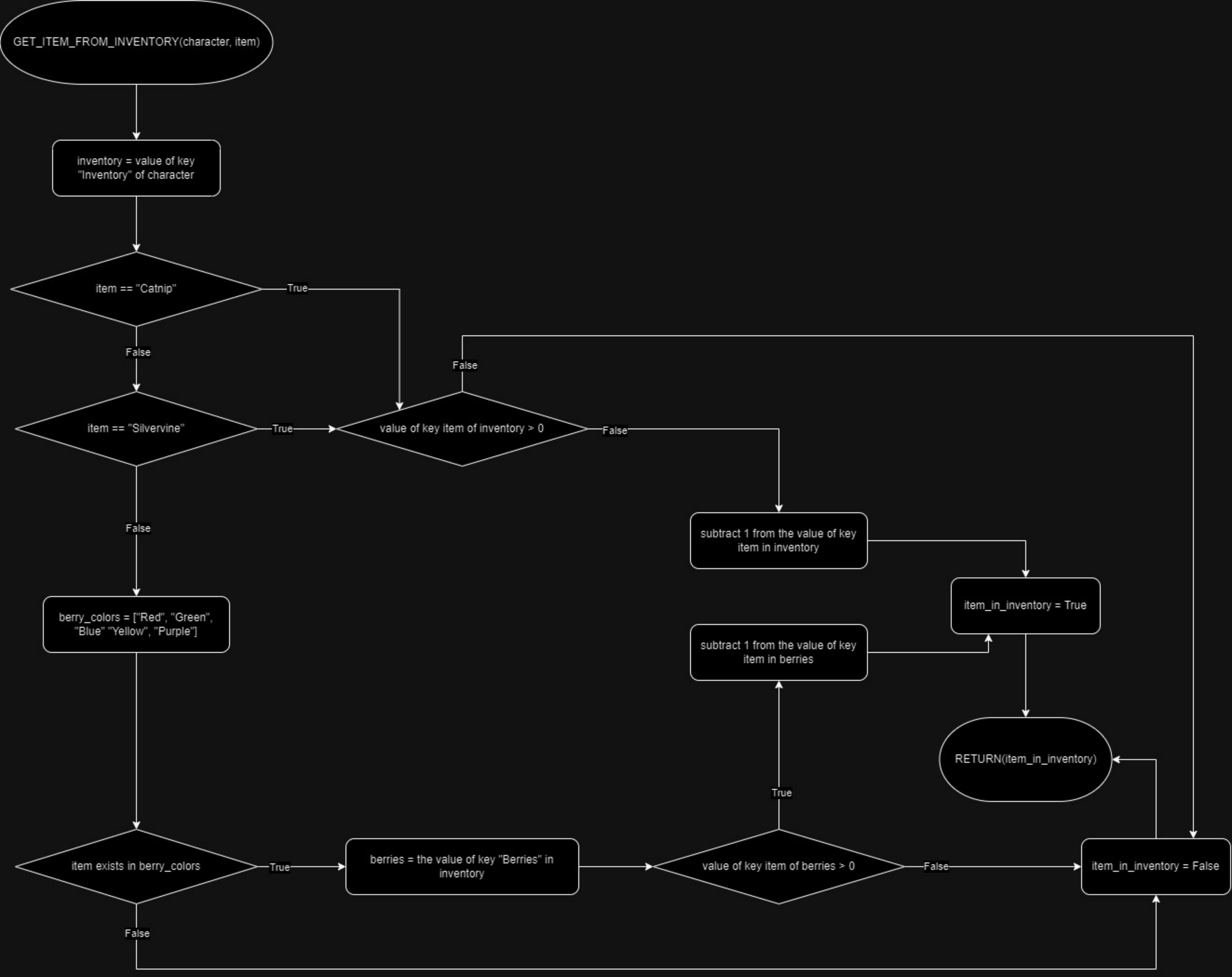

set key "Type" of new_entity to "Animal"

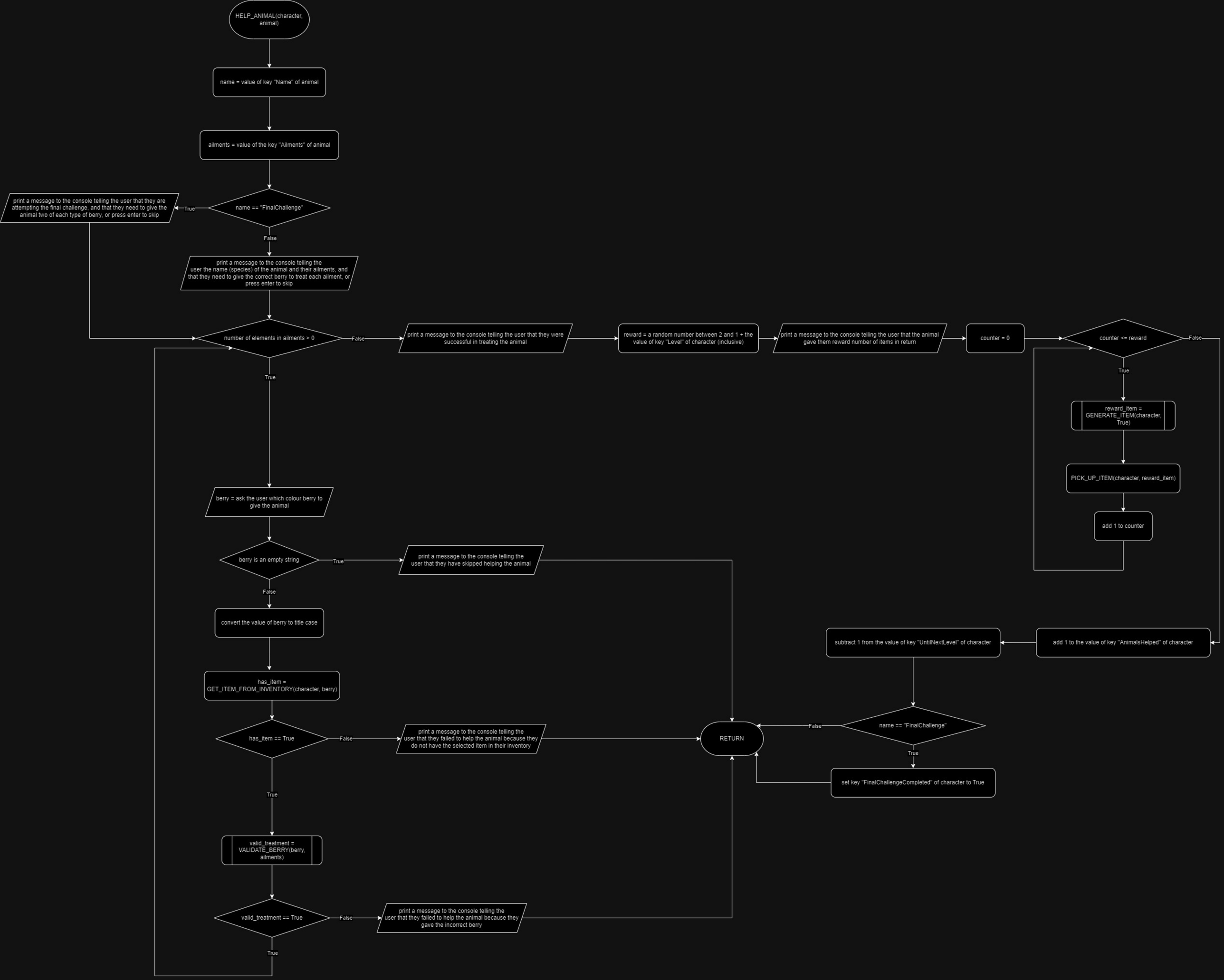

GENERATE_GROUND_BOARD()
ground_x_scale = 12
ground_y_scale = 12
ground_board = GENERATE_BOARD(-ground_x_scale, ground_x_scale, -ground_y_scale, ground_y_scale)
POPULATE_BOARD(ground_board, "TreeTrunk", 50)
RETURN(ground_board)

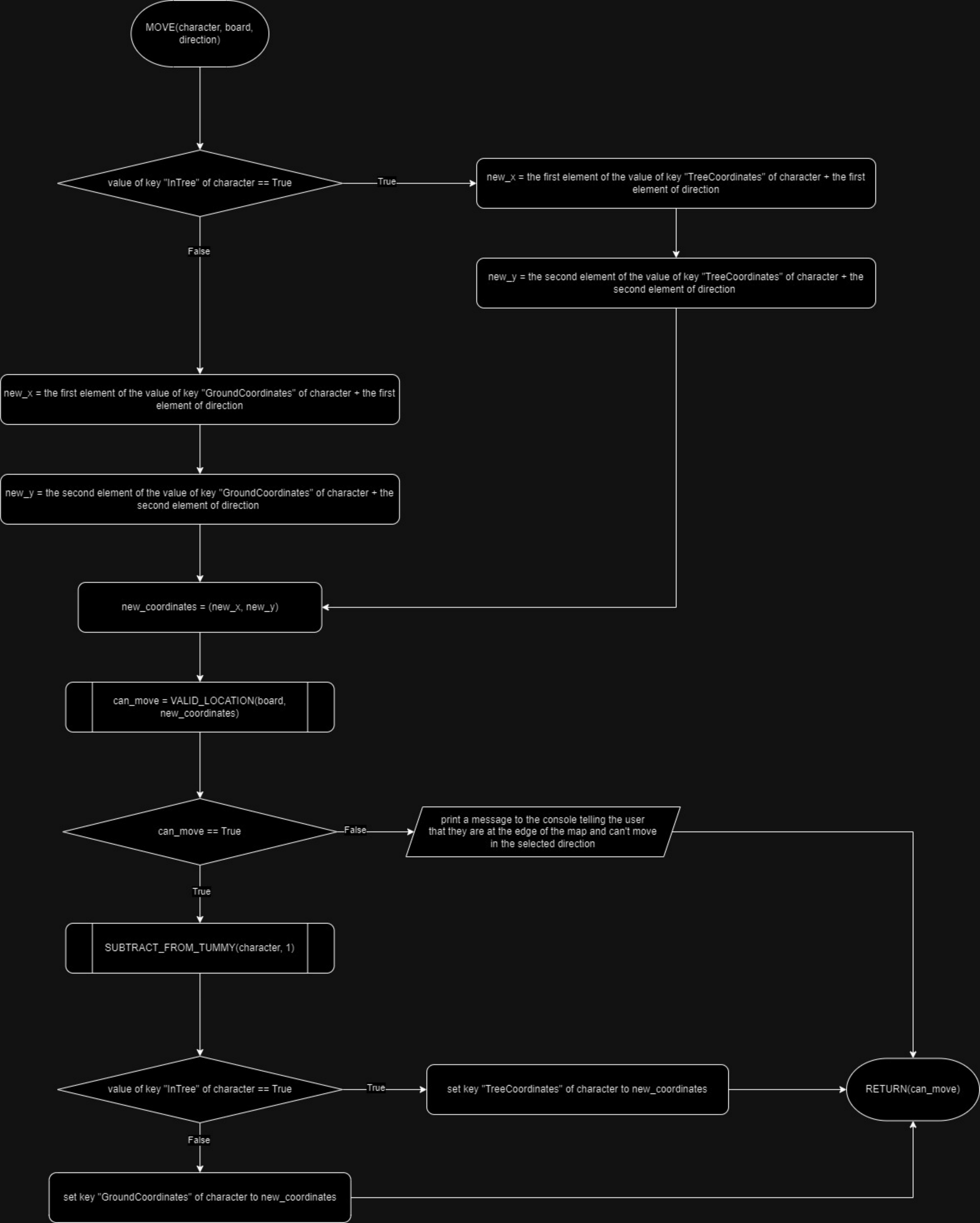

GENERATE_ITEM(character, always)
new_item = an empty dictionary
a random number between 1 and 100 (inclusive) == 100
True
set key "Name" of new_item to "Silvervine"
False
a random number between 1 and 20 (inclusive) == 20
True
set key "Name" of new_item to "Catnip"
False
always == True
True
False
key "InTree" of character == True
True
a random number between 1 and 5 (inclusive) == 5
True
False
False
new_item = None
set key "Name" of new_item to "Berry"
berry_color_options = ["Red", "Green", "Blue", "Yellow", "Purple"]
berry_color = randomly choose an element from berry_color_options
set key "Data" of new_item to berry_color
set key "Data" of new_item to None
set key "Type" of new_item to "Item"
RETURN(new_item)

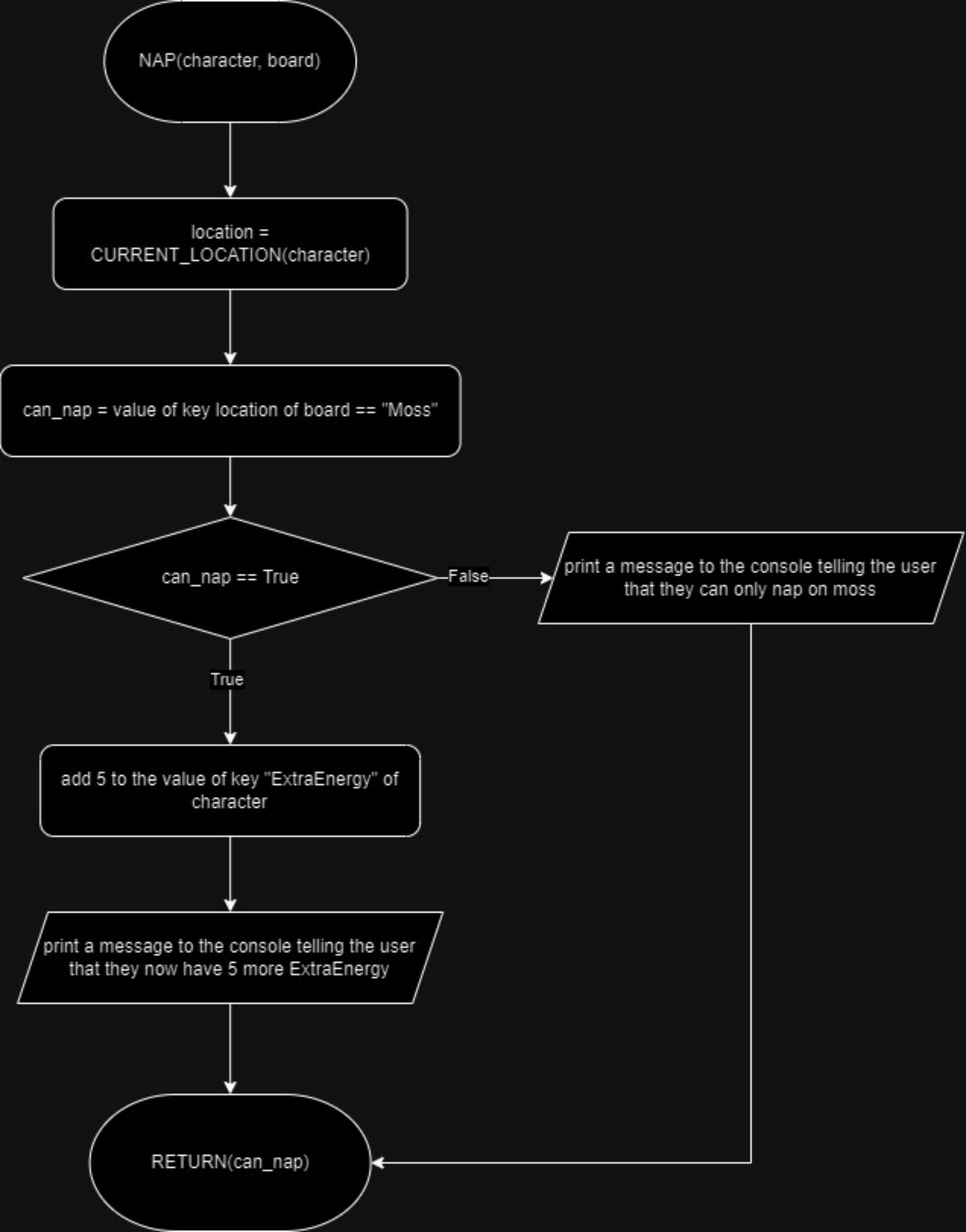

GENERATE_TREE_BOARD()
tree_scale_options = [1, 3, 5, 7, 9]
tree_scale = randomly choose an element from tree_scale_options
tree_board = GENERATE_BOARD(-tree_scale, tree_scale, -tree_scale, tree_scale)
set description of the coordinate (0, 0) of tree_board to "TreeTrunk"
POPULATE_BOARD(tree_board, "Moss", tree_scale)
RETURN(tree_board)

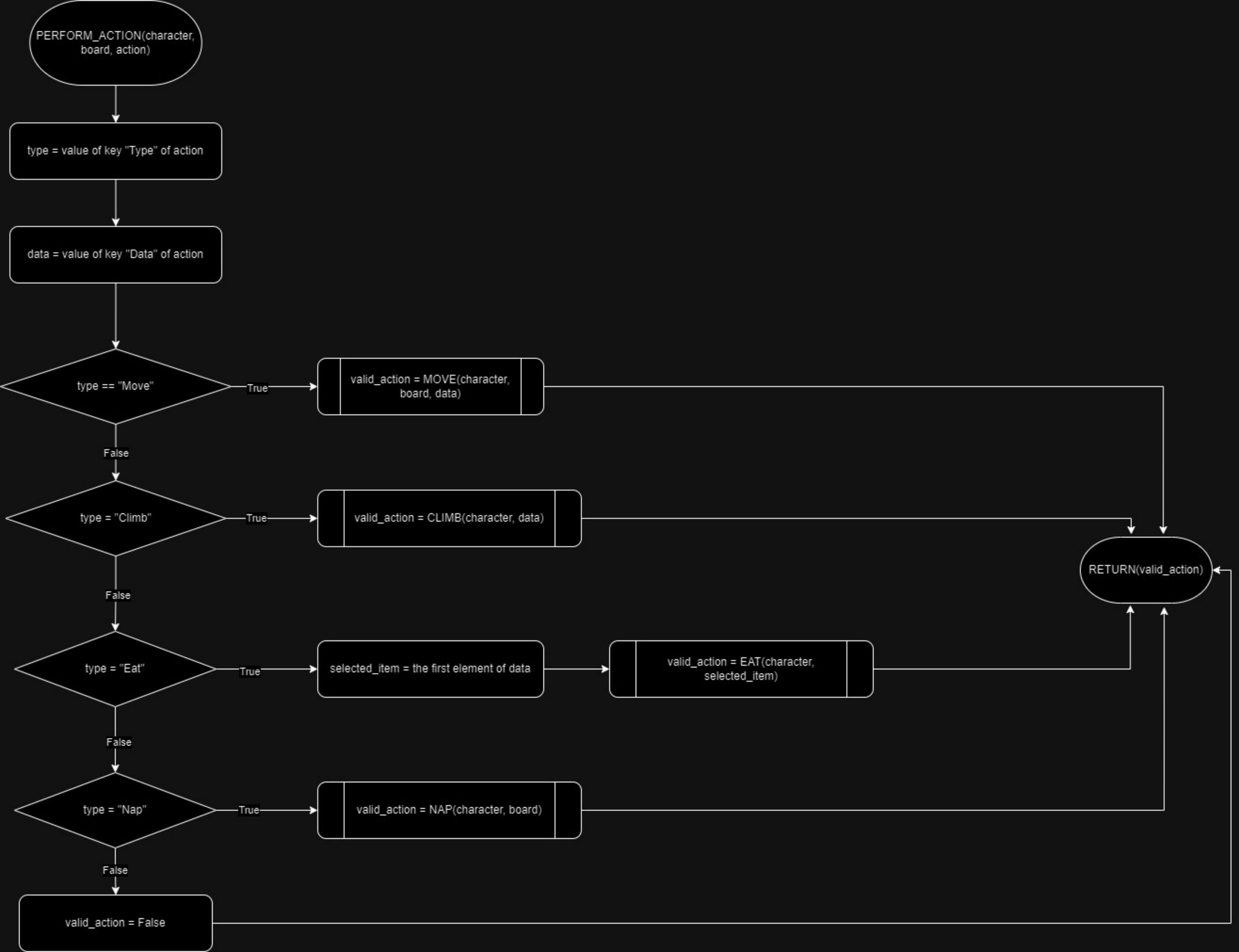

GET_ACTION_INPUT(character, board)
selected_action = ask user to enter an action
selected_action is an empty string
True
False
convert the text in selected_action to title case
action_tokens = split selected_action into a list of tokens
action = an empty dictionary
set key "Type" of action to the first element of action_tokens
set key "Data" of action to a slice of action_tokens from the second element to the last element
valid_actions = ["W", "A", "S", "D", "Climb", "Eat", "Nap", "Check", "Help"]
value of key "Type" of action exists in valid_actions
False
print a message to the console telling the user that the selected action is invalid
True
value of key "Type" of action == "Check"
True
valid_attributes = ["Tummy", "Level", "Inventory"]
False
value of key "Data" of action is an empty list
True
False
print a message to the console listing the available input options and what they do
True
value of key "Type" of action == "Help"
False
the first element of the value of key "Data" of action exists in valid_attributes
False
True
set key "Data" of action to (0, -1)
True
value of key "Type" of action == "W"
False
CHECK(character, board, action)
set key "Data" of action to (0, 1)
True
value of key "Type" of action == "S"
False
set key "Type" of action to "Move"
set key "Data" of action to (-1, 0)
True
value of key "Type" of action == "A"
False
set key "Data" of action to (1, 0)
True
value of key "Type" of action == "D"
False
RETURN(action)

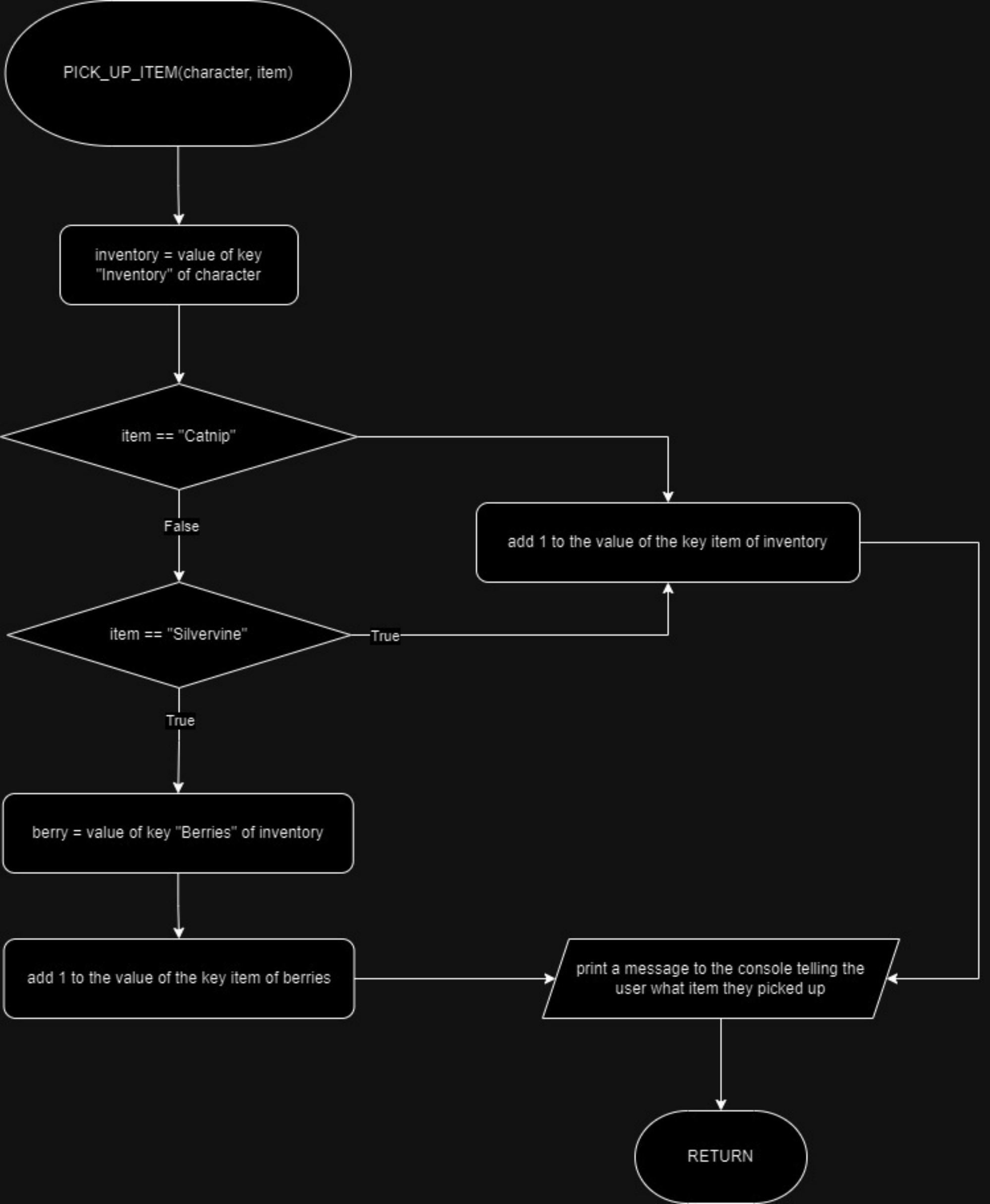

GET_ITEM_FROM_INVENTORY(character, item)
inventory = value of key "Inventory" of character
item == "Catnip"
True
False
item == "Silvervine"
True
False
value of key item of inventory > 0
False
True
subtract 1 from the value of key item in inventory
item_in_inventory = True
subtract 1 from the value of key item in berries
RETURN(item_in_inventory)
True
berry_colors = ["Red", "Green", "Blue" "Yellow", "Purple"]
item exists in berry_colors
True
False
berries = the value of key "Berries" in inventory
value of key item of berries > 0
False
item_in_inventory = False


HELP_ANIMAL(character, animal)
name = value of key "Name" of animal
ailments = value of the key "Ailments" of animal
name == "FinalChallenge"
True
print a message to the console telling the user that they are attempting the final challenge, and that they need to give the animal two of each type of berry, or press enter to skip
False
print a message to the console telling the user the name (species) of the animal and their ailments, and that they need to give the correct berry to treat each ailment, or press enter to skip
number of elements in ailments > 0
False
print a message to the console telling the user that they were successful in treating the animal
reward = a random number between 2 and 1 + the value of key "Level" of character (inclusive)
print a message to the console telling the user that the animal gave them reward number of items in return
counter = 0
counter <= reward
False
True
reward_item = GENERATE_ITEM(character, True)
PICK_UP_ITEM(character, reward_item)
add 1 to counter
True
berry = ask the user which colour berry to give the animal
berry is an empty string
True
print a message to the console telling the user that they have skipped helping the animal
False
convert the value of berry to title case
has_item = GET_ITEM_FROM_INVENTORY(character, berry)
has_item == True
False
print a message to the console telling the user that they failed to help the animal because they do not have the selected item in their inventory
True
valid_treatment = VALIDATE_BERRY(berry, ailments)
valid_treatment == True
False
print a message to the console telling the user that they failed to help the animal because they gave the incorrect berry
True
RETURN
add 1 to the value of key "AnimalsHelped" of character
subtract 1 from the value of key "UntilNextLevel" of character
name == "FinalChallenge"
False
True
set key "FinalChallengeCompleted" of character to True


MOVE(character, board, direction)
value of key "InTree" of character == True
True
False
new_x = the first element of the value of key "TreeCoordinates" of character + the first element of direction
new_y = the second element of the value of key "TreeCoordinates" of character + the second element of direction
new_x = the first element of the value of key "GroundCoordinates" of character + the first element of direction
new_y = the second element of the value of key "GroundCoordinates" of character + the second element of direction
new_coordinates = (new_x, new_y)
can_move = VALID_LOCATION(board, new_coordinates)
can_move == True
False
print a message to the console telling the user that they are at the edge of the map and can't move in the selected direction
True
SUBTRACT_FROM_TUMMY(character, 1)
value of key "InTree" of character == True
True
set key "TreeCoordinates" of character to new_coordinates
False
set key "GroundCoordinates" of character to new_coordinates
RETURN(can_move)


NAP(character, board)
location = CURRENT_LOCATION(character)
can_nap = value of key location of board == "Moss"
can_nap == True
False
print a message to the console telling the user that they can only nap on moss
True
add 5 to the value of key "ExtraEnergy" of character
print a message to the console telling the user that they now have 5 more ExtraEnergy
RETURN(can_nap)


PERFORM_ACTION(character, board, action)
type = value of key "Type" of action
data = value of key "Data" of action
type == "Move"
True
valid_action = MOVE(character, board, data)
False
type = "Climb"
True
valid_action = CLIMB(character, data)
False
type = "Eat"
True
selected_item = the first element of data
valid_action = EAT(character, selected_item)
False
type = "Nap"
True
valid_action = NAP(character, board)
False
valid_action = False
RETURN(valid_action)


PICK_UP_ITEM(character, item)
inventory = value of key "Inventory" of character
item == "Catnip"
False
add 1 to the value of the key item of inventory
item == "Silvervine"
True
True
berry = value of key "Berries" of inventory
add 1 to the value of the key item of berries
print a message to the console telling the user what item they picked up
RETURN

POPULATE_BOARD(board, name, times)

counter = 0

counter <= times

False

RETURN

False

x_coordinate = generate a random number between the minimum and maximum x coordinates on the board (inclusive)

y_coordinate = generate a random number between the minimum and maximum y coordinates on the board (inclusive)

coordinate = the tuple (x_coordinate, y_coordinate)

coordinate == (0, 0)

True

False

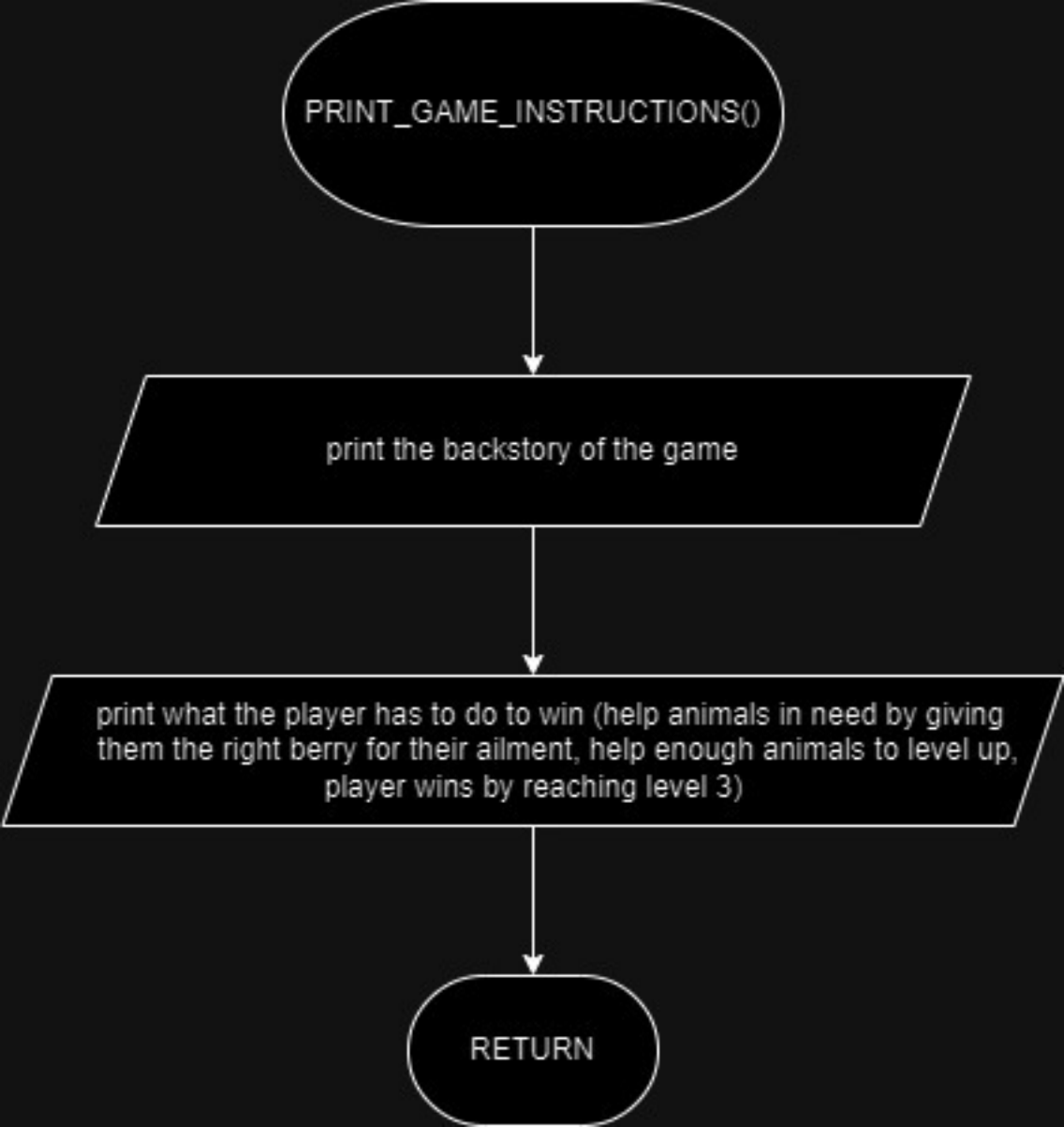
description of the coordinate on the board is None

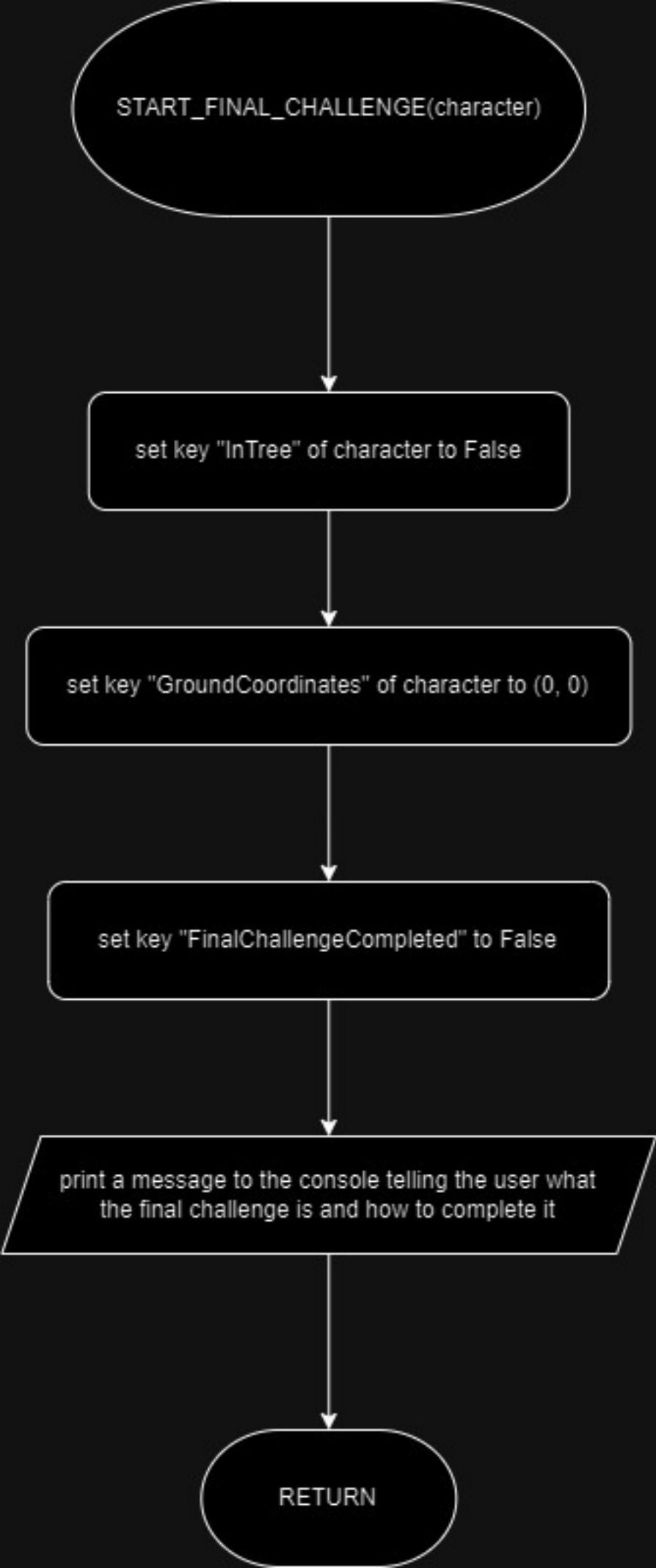
False

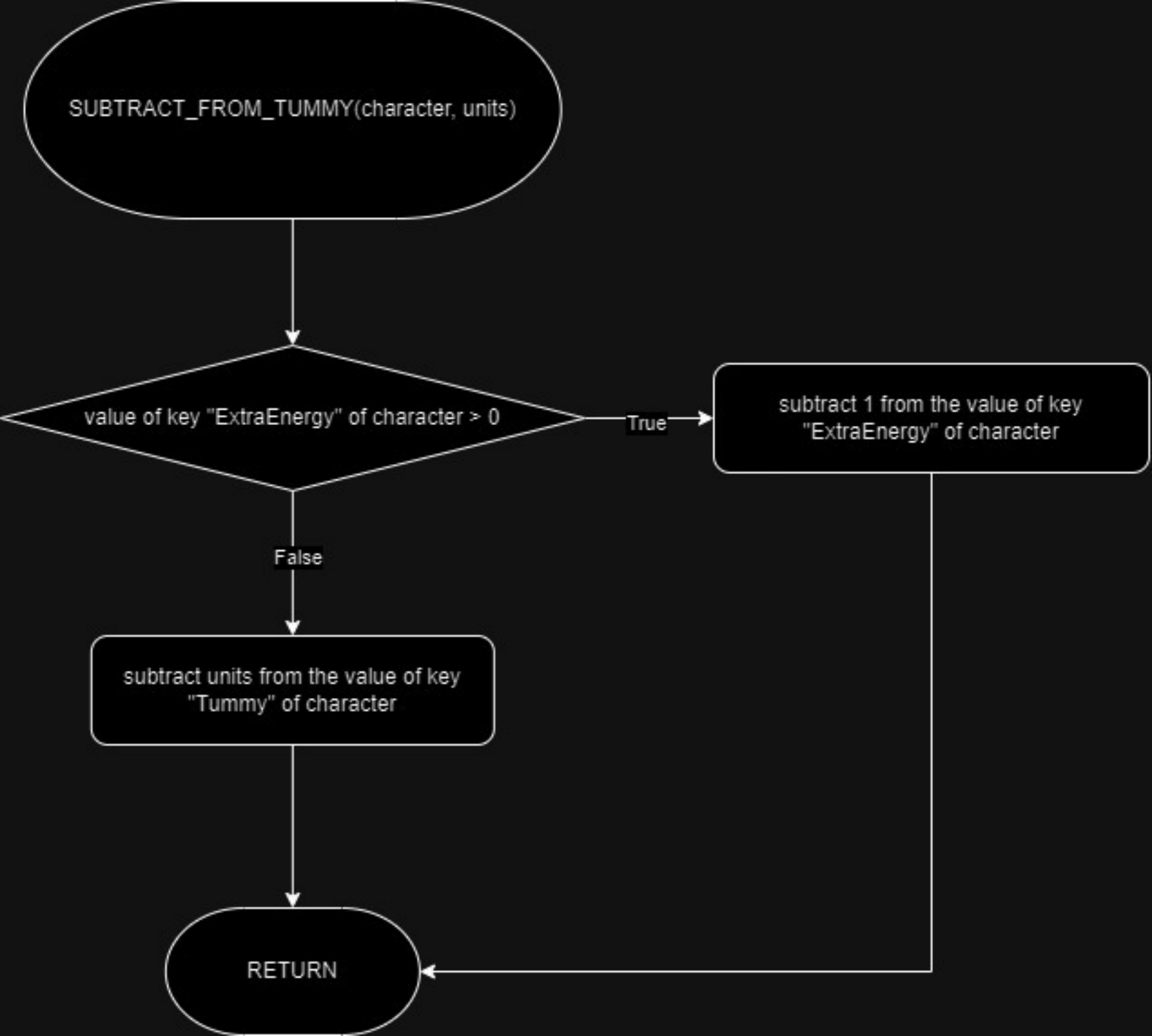
True

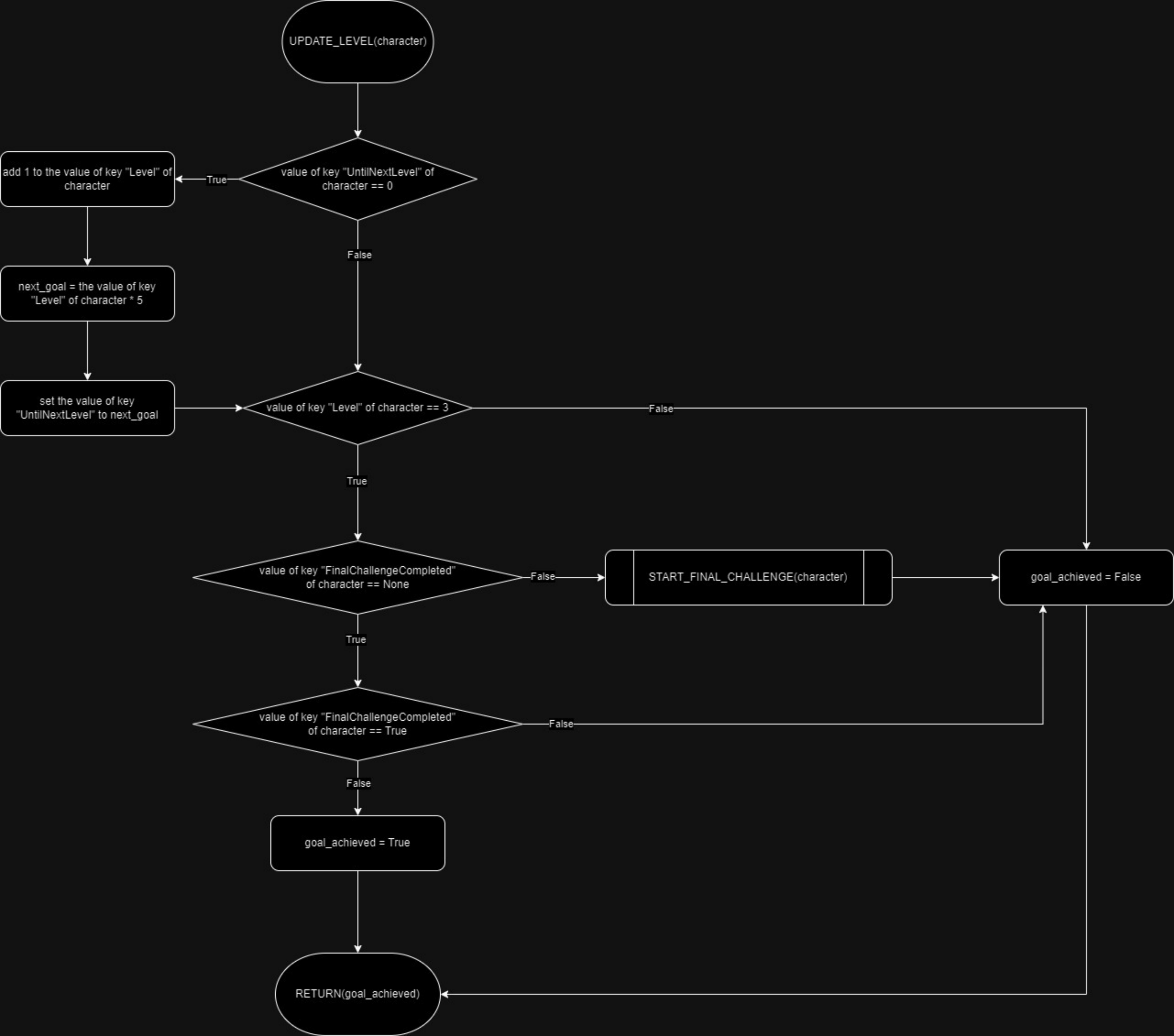
set description of coordinate on the board to name

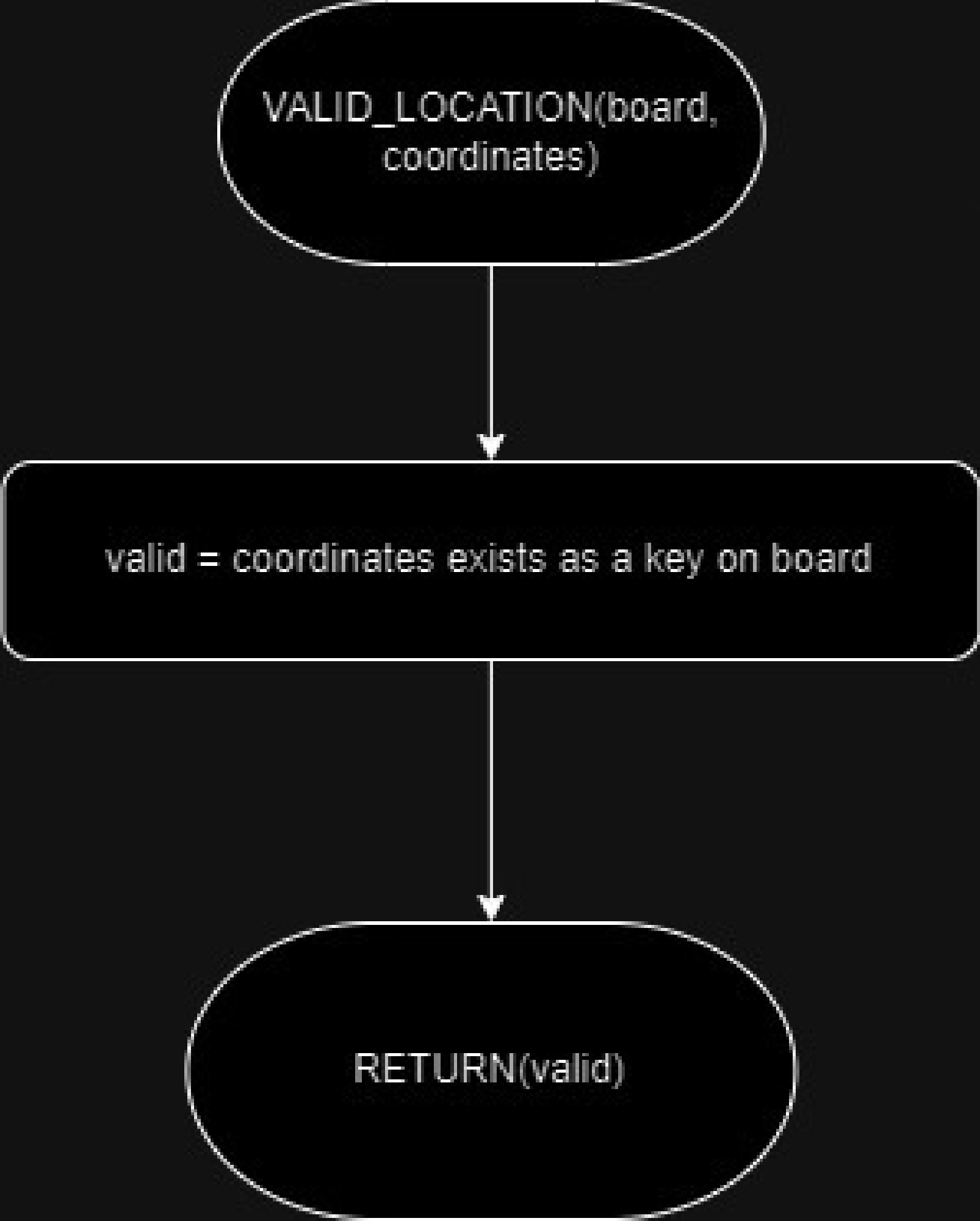
add 1 to counter


PRINT_GAME_INSTRUCTIONS()
print the backstory of the game
print what the player has to do to win (help animals in need by giving them the right berry for their ailment, help enough animals to level up, player wins by reaching level 3)
RETURN


START_FINAL_CHALLENGE(character)
set key "InTree" of character to False
set key "GroundCoordinates" of character to (0, 0)
set key "FinalChallengeCompleted" to False
print a message to the console telling the user what the final challenge is and how to complete it
RETURN


SUBTRACT_FROM_TUMMY(character, units)
value of key "ExtraEnergy" of character > 0
True
subtract 1 from the value of key "ExtraEnergy" of character
False
subtract units from the value of key "Tummy" of character
RETURN


UPDATE_LEVEL(character)
value of key "UntilNextLevel" of character == 0
True
add 1 to the value of key "Level" of character
False
next_goal = the value of key "Level" of character * 5
set the value of key "UntilNextLevel" to next_goal
value of key "Level" of character == 3
False
True
value of key "FinalChallengeCompleted" of character == None
False
START_FINAL_CHALLENGE(character)
goal_achieved = False
True
value of key "FinalChallengeCompleted" of character == True
False
False
goal_achieved = True
RETURN(goal_achieved)


VALID_LOCATION(board, coordinates)
valid = coordinates exists as a key on board
RETURN(valid)