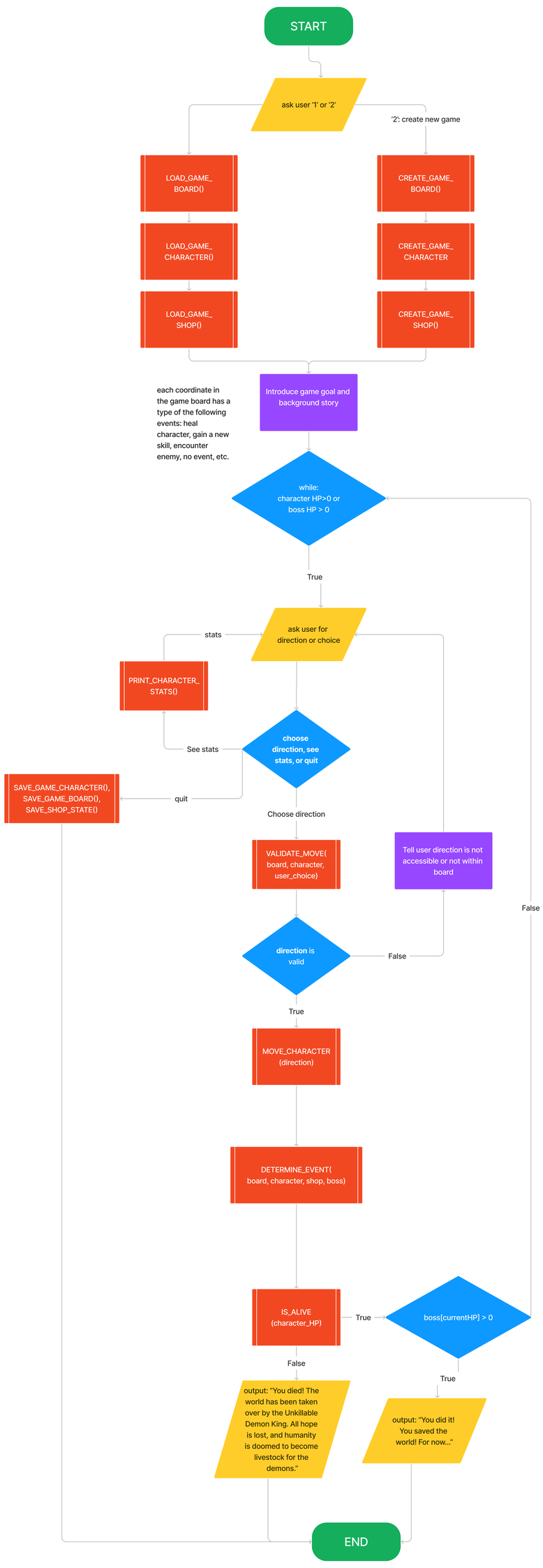
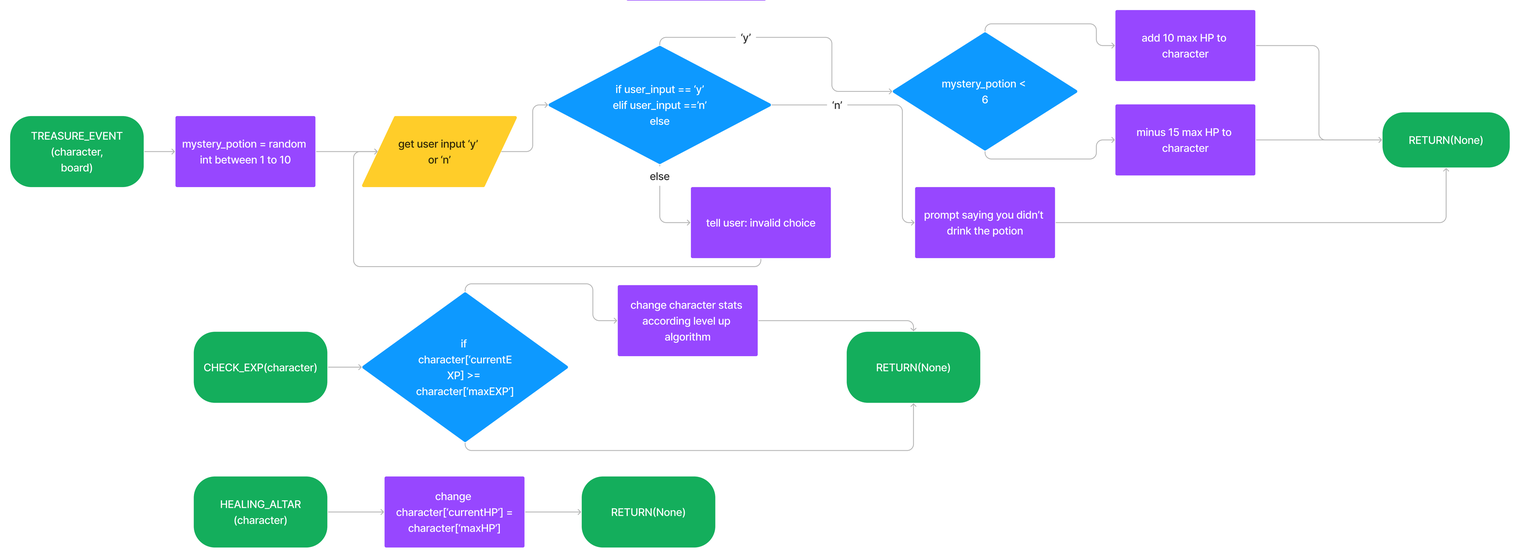
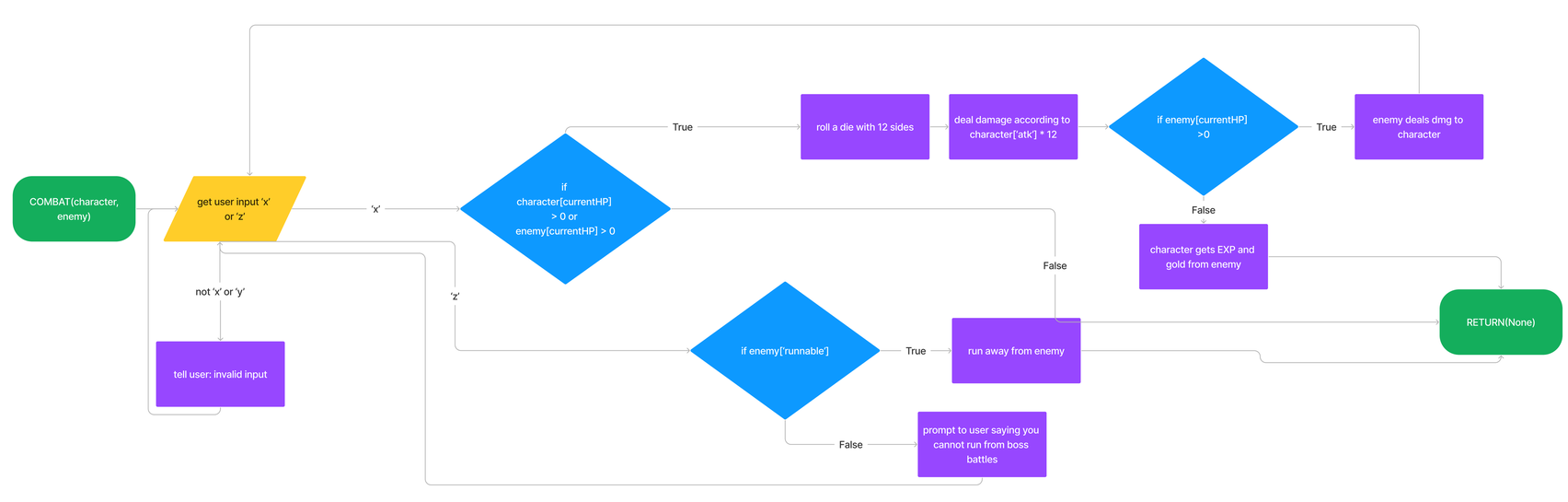
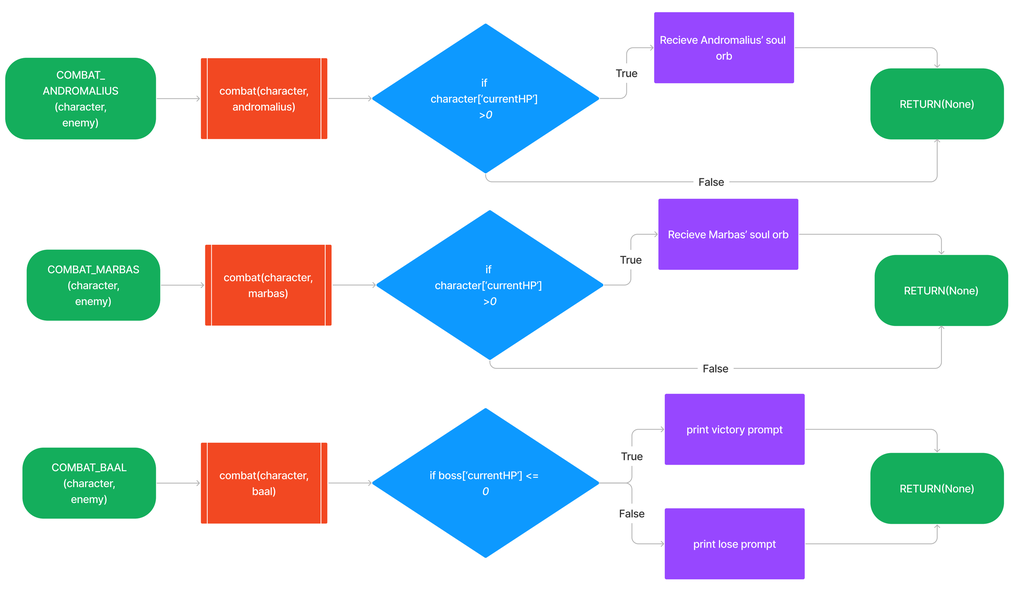
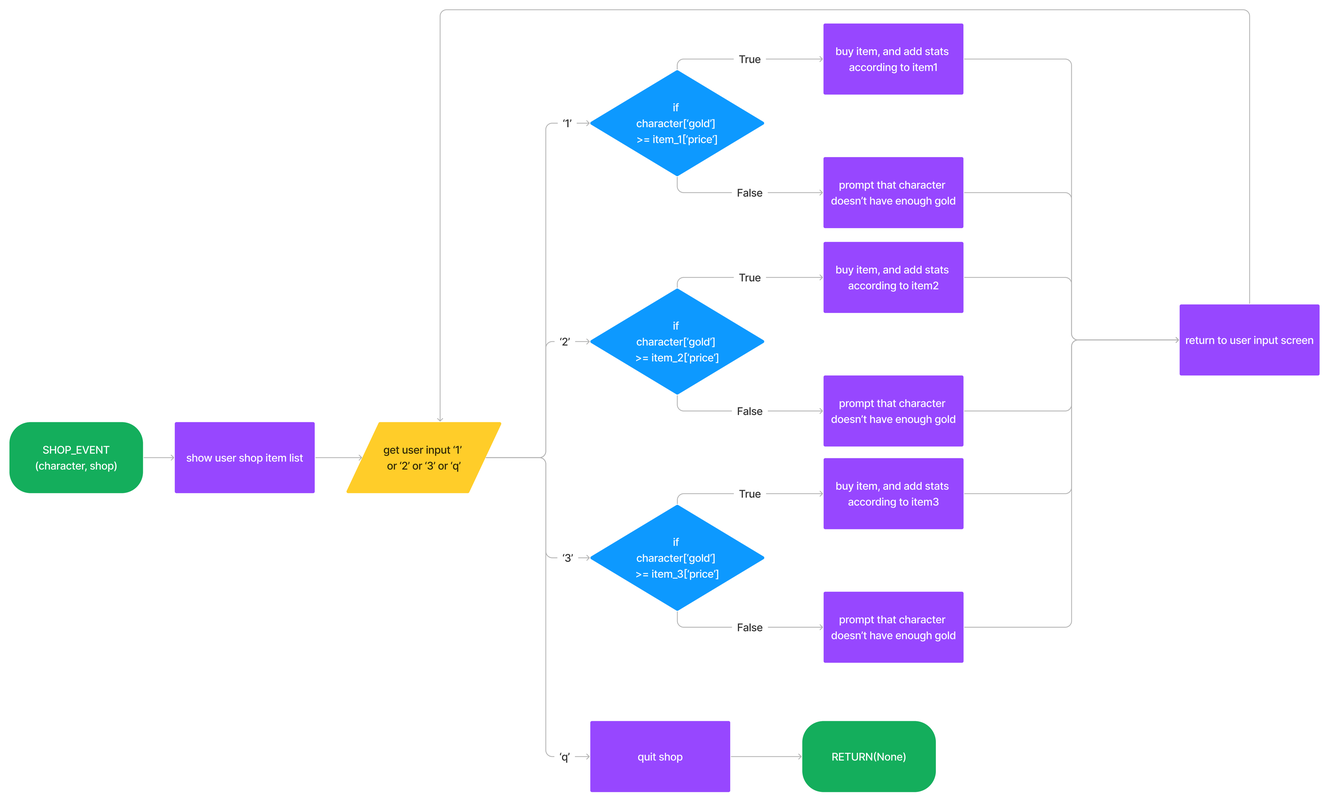

START
ask user '1' or '2'
'2': create new game
LOAD_GAME_BOARD()
LOAD_GAME_CHARACTER()
LOAD_GAME_SHOP()
CREATE_GAME_BOARD()
CREATE_GAME_CHARACTER
CREATE_GAME_SHOP()
each coordinate in the game board has a type of the following events: heal character, gain a new skill, encounter enemy, no event, etc.
Introduce game goal and background story
while: character HP>0 or boss HP > 0
True
ask user for direction or choice
stats
PRINT_CHARACTER_STATS()
See stats
choose direction, see stats, or quit
quit
SAVE_GAME_CHARACTER(), SAVE_GAME_BOARD(), SAVE_SHOP_STATE()
Choose direction
VALIDATE_MOVE(board, character, user_choice)
Tell user direction is not accessible or not within board
direction is valid
False
True
MOVE_CHARACTER (direction)
DETERMINE_EVENT(board, character, shop, boss)
IS_ALIVE (character_HP)
True
boss[currentHP] > 0
False
True
output: "You died! The world has been taken over by the Unkillable Demon King. All hope is lost, and humanity is doomed to become livestock for the demons."
output: "You did it! You saved the world! For now..."
END
False

MAKE_BOARD()
Introduce ascii art mini-map created out of list of strings
create variable map_dictionary = {}
enumerate through list of string and store tuples of coordinates as keys and tiles as values
RETURN(board)
IS_ALIVE(character)
character is above 0 hp
True
False
character is above 0HP
character is less than 0HP
RETURN(bool)
PRINT_BOARD(board, character)
iterate through dictionary, and print the value associated with the key
print "O" at the character's current location
RETURN(None)
SLOW_PRINT (text, delay=0.01)
print each character in the text
add time.sleep(delay) after each character
RETURN(None)
VALIDATE_MOVE(board, character, user_direction)
user given direction is accessible(a whitespace, or event tile)
True
False
coordinate exists
coordinate does not exist, or is 'X'
RETURN(bool)
MOVE_CHARACTER (character, user_direction)
move character to desired location
RETURN(None)
DICT_TO_JSON_CONVERSION(board)
board keys are tuples
True
False
Convert keys to strings
Convert keys to tuple
RETURN(dict)
SAVE_GAME_CHARACTER (character)
Dump current character dictionary into a json file
RETURN(None)
SAVE_GAME_BOARD (board)
Dump current board dictionary into a json file
convert board keys from tuples to string using dict_to_json_string_conversion (board)
RETURN(None)
SAVE_GAME_SHOP (shop)
Dump current shop dictionary into a json file
RETURN(None)
LOAD_GAME_CHARACTER()
Load json file into a dictionary variable
RETURN(character)
LOAD_GAME_BOARD()
Load board json file into a variable
convert string keys back into tuples using dict_to_json_string_conversion(board)
RETURN(game_board)
LOAD_GAME_SHOP()
Load shop json file into a dictionary variable
RETURN(shop)
CHANCE_OF_EVENT (probability)
create random float between 0 and 1
True
False
random_float < probability
random_float < probability
RETURN(bool)
CHECK_IF_LV_10 (character)
character["level"] < 10
True
False
prompt that says you cannot pass
character['y-position'] + 1
prompt that says you cannot pass
RETURN(bool)
CHECK_FOR_ORBS (character)
character["orb"] < 2
True
False
prompt that says you cannot pass
character['y-position'] + 1
prompt that says you can pass
RETURN(None)
MAKE_CHARACTER()
create a character dictionary containing character information
RETURN(character)
MAKE_PRE_LV_10_MOBS()
create a list of monster dictionary containing monster information
RETURN(list_of_mobs)
MAKE_POST_LV_10_MOBS()
create a list of monster dictionary containing monster information
RETURN(list_of_mobs)
MAKE_MARBAS()
create a mid-boss dictionary containing mid-boss information
RETURN(marbas)
MAKE_ANDROMALIUS()
create a mid-boss dictionary containing mid-boss information
RETURN(andromalius)
MAKE_BAAL()
create a final-boss dictionary containing final-boss information
RETURN(baal)
MAKE_SHOP()
create a shop list of dictionary containing items that are purchasable
RETURN(shop_items)


TREASURE_EVENT (character, board)
mystery_potion = random int between 1 to 10
get user input 'y' or 'n'
if user_input == 'y' elif user_input =='n' else
'y'
'n'
else
tell user: invalid choice
mystery_potion < 6
add 10 max HP to character
minus 15 max HP to character
prompt saying you didn't drink the potion
RETURN(None)
CHECK_EXP(character)
if character['currentEXP] >= character['maxEXP']
change character stats according level up algorithm
RETURN(None)
HEALING_ALTAR (character)
change character['currentHP'] = character['maxHP']
RETURN(None)


COMBAT(character, enemy)
get user input 'x' or 'z'
'x'
'z'
not 'x' or 'y'
tell user: invalid input
if character[currentHP] > 0 or enemy[currentHP] > 0
True
roll a die with 12 sides
deal damage according to character['atk'] * 12
if enemy[currentHP] >0
True
enemy deals dmg to character
False
character gets EXP and gold from enemy
False
if enemy['runnable']
True
run away from enemy
False
prompt to user saying you cannot run from boss battles
RETURN(None)


COMBAT_ANDROMALIUS (character, enemy)
combat(character, andromalius)
if character['currentHP'] >0
True
Recieve Andromalius' soul orb
RETURN(None)
False
COMBAT_MARBAS (character, enemy)
combat(character, marbas)
if character['currentHP'] >0
True
Recieve Marbas' soul orb
RETURN(None)
False
COMBAT_BAAL (character, enemy)
combat(character, baal)
if boss['currentHP'] <= 0
True
print victory prompt
RETURN(None)
False
print lose prompt


SHOP_EVENT (character, shop)
show user shop item list
get user input '1' or '2' or '3' or 'q'
'1'
'2'
'3'
'q'
if character['gold'] >= item_1['price']
True
buy item, and add stats according to item1
False
prompt that character doesn't have enough gold
if character['gold'] >= item_2['price']
True
buy item, and add stats according to item2
False
prompt that character doesn't have enough gold
if character['gold'] >= item_3['price']
True
buy item, and add stats according to item3
False
prompt that character doesn't have enough gold
quit shop
RETURN(None)
return to user input screen