Hazel Jeong
A01166162

```
START
```

Ask the user for a character name

Store the character name

GAME_INSTRUCTIONS
(character name)

character =
MAKE_CHARACTER
(character name)

skills =
DEFINE_SKILLS()

dishes =
DEFINE_DISHES()

dish_options =
SHOW_DISH_OPTIONS
(dishes)

Print out dish_options

dish_choice =
GET_USER_DISH_CHOICE
(dishes)

ingredients =
INGREDIENTS_TO_GET
(character, dish_choice)

rows = 5

columns = 5

The character's HP is bigger than 0 — False → Print out a message indicating the character is dead → END

True

The user has achieved the final goal of the game — True → Print out a congratulation message → END

False

DESCRIBE_CURRENT_LOCATION
(character)

direction =
GET_USER_
DIRECTION_CHOICE()

valid_move =
VALIDATE_MOVE
(rows, columns,
character, direction)

valid_move is True — False → Print out a warning message that the user can't move that way

True

MOVE_CHARACTER
(character, direction)

there_is_a_battle =
CHECK_FOR_BATTLES(character)

There's a battle — False

True

BATTLE
(character,
ingredients,
skills)

The character has leveled up — False

True

LEVEL_UP_PROTOCOL
(character, skills)

GAME_INSTRUCTIONS
(character name)

↓

Print out greetings
with the character name

↓

Print out game
instructions

↓

RETURN

MAKE_CHARACTER
(character name)

↓

Create a data structure called
'character', which can hold the
following nested key-value pairs;
Name: character name,
X-coordinate: 2,
Y-coordinate: 2,
Current HP: 5,
Max HP: 5,
Level: 1,
Skills: {1: Rinse},
Items: an empty set

↓

RETURN(character)

DEFINE_SKILLS()

↓

Create a data structure called
'skills', which can hold the
following nested key-value pairs;
Rinse: {Hit: 2, Accuracy: 0.8}
Cut: {Hit: 5, Accuracy: 0.5}
Fire: {Hit: 10, Accuracy: 0.2}

↓

RETURN(skills)

DEFINE_DISHES()

↓

Create a data structure called 'dishes', which can hold the following nested key-value pairs;
Galbijjim: {Difficulty: 5, Vegetables: (Carrots, Potatoes, Onions, Garlic), Meat: Beef},
Jeyuk: {Difficulty: 3, Vegetables: (Carrots, Onions, Garlic), Meat: Pork},
Dakgalbi: {Difficulty: 2, Vegetables: (Cabbages, Potatoes, Onions, Garlic), Meat: Chicken},
Kongbul: {Difficulty: 3, Vegetables: (Bean Sprouts, Onions, Garlic), Meat: Pork},
Jjimdak: {Difficulty: 2, Vegetables: (Carrots, Potatoes, Onions, Garlic), Meat: Chicken},
Samgyeopsal: {Difficulty: 1, Vegetables: (Lettuce, Garlic), Meat: Pork},
Bulgogi: {Difficulty: 4, Vegetables: (Carrots, Onions, Garlic), Meat: Beef}

↓

RETURN(dishes)

SHOW_DISH_OPTIONS
(dishes)

↓

Create a string called
'dish_options' containing a
numbered list of dish name
and difficulty pairs from the
input parameter 'dishes'

↓

RETURN(dish_options)

GET_USER_DISH_CHOICE
(dishes)

↓

Ask the user
what dish they
would like to cook
in the game

↓

Store ingredients of the selected
dish into a data structure can hold
key-value pairs below;
1: list of vegetable ingredients,
2: string of meat ingredient

↓

RETURN(dish_choice)

INGREDIENTS_TO_GET
(character, dish_choice)

↓

Create a variable called 'ingredients'
containing a set of ingredients that
matches to the character's level.
For example, if the character level is
1, ingredients will be the value
matches to the key '1' in
'dish_choice').

↓

RETURN(ingredients)

DESCRIBE_CURRENT_LOCATION
(character)

↓

Print out the current coordinate
of the character and the current
HP from the input parameter
'character'

↓

RETURN

Hazel Jeong
A01166162

GET_USER_DIRECTION_CHOICE()

Ask the user for the direction they wish to travel with a numbered list of directions (1: North, 2: South, 3: East, 4: West)

The user input is valid — False

True

Store the direction the user chose to 'direction'

RETURN(direction)

MOVE_CHARACTER (character, direction)

direction == 1 — True → character's Y-coordinate -= 1

False

direction == 2 — True → character's Y-coordinate += 1

False

direction == 3 — True → character's X-coordinate += 1

False

direction == 4 — True → character's X-coordinate -= 1

RETURN

VALIDATE_MOVE (rows, columns, character, direction)

direction == 1 — True → character's Y-coordinate > 0

False

direction == 2 — True → character's Y-coordinate < rows - 1

False

direction == 3 — True → character's X-coordinate < columns - 1

False

direction == 4 — True → character's X-coordinate > 0

True → Assign True to valid_move

False → Assign False to valid_move

RETURN(valid_move)

Hazel Jeong
A01166162

CHECK_FOR_BATTLES (character)

The character's level is 3

- True → Assign True to 'there_is_a_battle' variable with 100% of chance
- False → Assign True to 'there_is_a_battle' variable with 50% of chance

RETURN(there_is_a_battle)

LEVEL_UP_PROTOCOL (character, skills)

Set the following values in 'character';
Level += 1,
Max HP *= 3,
Current HP = Max HP
X-coordinate = 2
Y-coordinate = 2

Add second skill from 'skills' to character's Skills

Empty the character's Items

RETURN

Hazel Jeong
A01166162

BATTLE
(character, ingredients, skills)

The character's level is 3 —True→ Assign the final dish's HP, Hit, and Accuracy with the following values;
HP = 20,
Hit = 5,
Accuracy = 0.8

False

Among the ingredients the user needs to collect for the current character's level, select an item randomly

Print out what ingredient the user encountered

Print out the skills the user currently have

The character's level is 1 —True→ Assign the picked ingredient's HP, Hit, and Accuracy with the following values;
HP = 5,
Hit = 1,
Accuracy = 0.5

False

The picked ingredient is beef —True→ Assign the picked ingredient's HP, Hit, and Accuracy with the following values;
HP = 10,
Hit = 4,
Accuracy = 0.8

False

The picked ingredient is pork —True→ Assign the picked ingredient's HP, Hit, and Accuracy with the following values;
HP = 10,
Hit = 3,
Accuracy = 0.6

False

Assign the picked ingredient's HP, Hit, and Accuracy with the following values;
HP = 10,
Hit = 2,
Accuracy = 0.6

The ingredient's HP is bigger than 0

False

True

The ingredient's attack hit the character

False

True

Reduce the character's HP by the Hit amount of the ingredient

Ask the user what skill they wish to use for the turn

The user input is valid

False

True

The attack hit the ingredient

False

True

Reduce the ingredient's HP by the Hit amount of the skill used

Print out a message that the character has obtained the ingredient

Add the ingredient to the character's Items

RETURN