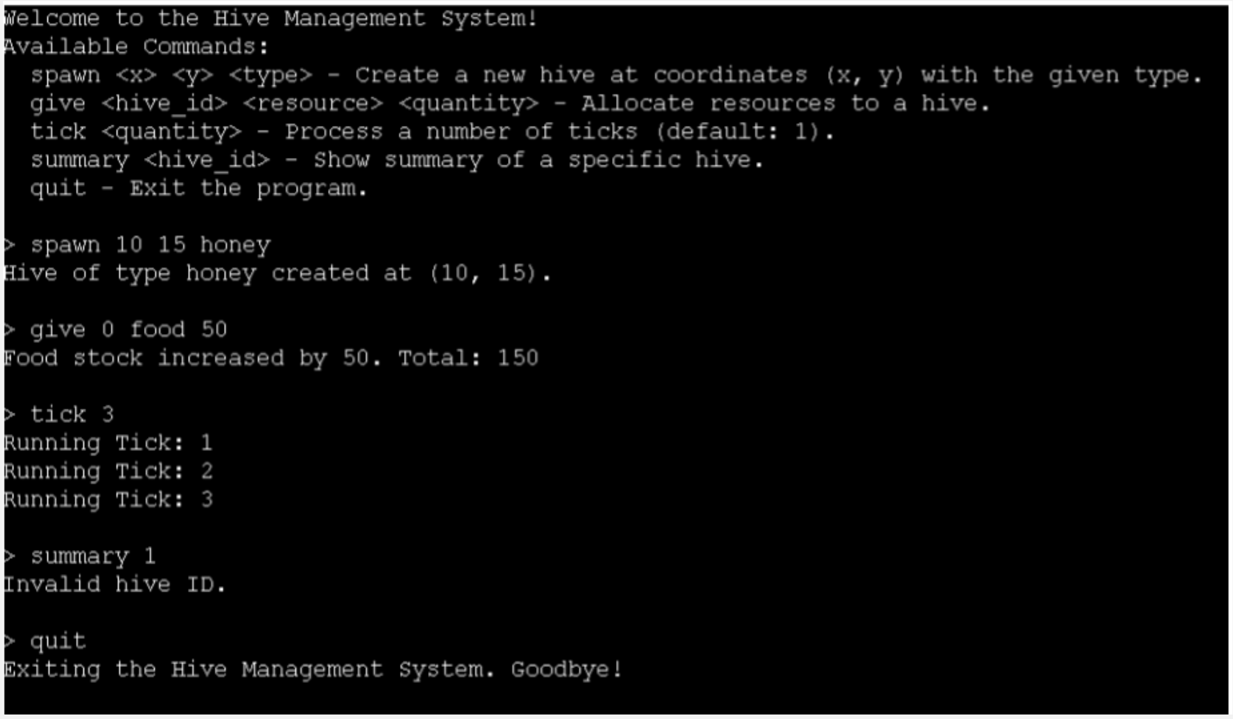
SE ASSIGNMENT

NAME: FATIMA MARAYAM

ROLL NUMBER: 27

SCREEN SHOT



```
Welcome to the Hive Management System!
Available Commands:
  spawn <x> <y> <type> - Create a new hive at coordinates (x, y) with the given type.
  give <hive_id> <resource> <quantity> - Allocate resources to a hive.
  tick <quantity> - Process a number of ticks (default: 1).
  summary <hive_id> - Show summary of a specific hive.
  quit - Exit the program.

> spawn 10 15 honey
Hive of type honey created at (10, 15).

> give 0 food 50
Food stock increased by 50. Total: 150

> tick 3
Running Tick: 1
Running Tick: 2
Running Tick: 3

> summary 1
Invalid hive ID.

> quit
Exiting the Hive Management System. Goodbye!
```

EXPLANATION OF TASK:

Room Class: Represents individual rooms in a hive with attributes room name and room capacity. Provides a constructor for initializing room details.

Bug Abstract Class:

Serves as the base class for all bugs with a pure virtual function show traits (), ensuring derived classes implement this method.

Hive Class:

Encapsulates a hive's properties, including hive_type, coordinates (coord_x, coord_y), food_stock, leader alive status, room_list, and bug_list. Methods include adding rooms and bugs, gathering resources, simulating tick-based changes, and displaying hive information. Resource Management: gather_resources() increases food_stock for the hive when "food" is gathered and handles invalid resource types. During each tick (process_tick), the hive's food stock decreases by 5 units. If it reaches zero, the leader dies, and a message is displayed.

Field Singleton Class:

Implements the singleton pattern to manage the game world, ensuring only one instance of the field class exists. Stores all hives (hive_list) and provides methods to create hives, allocate resources, process game ticks, and display hive summaries. Field's Singleton Implementation: Uses a static pointer field_instance to hold the single instance. The get_instance() method ensures thread-safe, lazy initialization of the field instance. Bug Traits and Decorator Pattern:basic_bug defines the simplest type of bug with basic traits.strong_bug and smart_bug extend a base bug using the decorator pattern, adding "Strong" and "Smart" traits dynamically.

Bug Factory:

Implements a factory pattern with the make_bug method to create different types of bugs (worker_bug and fighter_bug).Uses decorators to enrich bugs, like adding the "Strong" trait to fighter_bug. Dynamic Memory Management: Uses std::shared_ptr to manage memory dynamically for objects such as rooms, bugs, and hives, avoiding memory leaks. Command Interface: Implements a simple command-line interface for users to interact with the hive management system using commands like spawn, give, tick, summary, and quit. Spawn Command:Creates a new hive at specified coordinates with a given type and adds it to the field. Give Command:Allocates a specified resource (e.g., food) to a hive identified by its ID, ensuring valid IDs and quantities.

Tick Command:

Processes a specified number of ticks, with a default of 1 tick, simulating the consumption of resources and other time-based changes. Summary Command:Displays detailed information about a specific hive, including its type, coordinates, food stock, leader status, room count, and bug count. Program Flow:Continuously reads user commands in a loop until quit is entered.Parses input using std::istringstream for robust command handling, ensuring valid input formats before executing commands.