

a) The Abstract Factory is going to be responsible for spawning enemies. The easiest way to set this up is to create an abstract base class that defines the interface for the factory, as follows:

```
from abc import ABC, abstractmethod

class EnemyFactory(ABC):
    @abstractmethod
    def spawn(self) -> Enemy:
        pass
```

Now we can create subclasses of `EnemyFactory` that implement the `spawn` method. For example, the `EasyEnemyFactory` class can be implemented as follows:

```
class EasyEnemyFactory(EnemyFactory):
    def spawn(self) -> Enemy:
        enemy_type = random.choice([EnemyType.KNIGHT, EnemyType.ARCHER])
        health = random.randint(30, 60)
        attack_power = random.randint(20, 40)
        defense = random.randint(10, 20)
        return Enemy(enemy_type, health, attack_power, defense)
```

Similarly you can define the `MediumEnemyFactory` and `HardEnemyFactory` classes. See the full code in the `solution2.py` file.

b) A more functional approach to creating enemies based on the type of spawn point would be to use a dictionary that maps the spawn point type to a function that creates the enemy. For example:

```
def easy_spawn() -> Enemy:
    enemy_type = random.choice([EnemyType.KNIGHT, EnemyType.ARCHER])
    health = random.randint(30, 60)
    attack_power = random.randint(20, 40)
    defense = random.randint(10, 20)
    return Enemy(enemy_type, health, attack_power, defense)

def medium_spawn() -> Enemy:
    # code for medium spawn

def hard_spawn() -> Enemy:
    # code for hard spawn

class SpawnType(StrEnum):
    EASY = auto()
    MEDIUM = auto()
    HARD = auto()

SPAWN_FUNCTIONS = {
    SpawnType.EASY: easy_spawn,
```

```
SpawnType.MEDIUM: medium_spawn,  
SpawnType.HARD: hard_spawn,  
}
```

Now you can use the dictionary to create enemies based on the spawn point type:

```
def spawn_enemies(spawn_type: SpawnType, count: int) -> list[Enemy]:  
    spawn_function = SPAWN_FUNCTIONS[spawn_type]  
    return [spawn_function() for _ in range(count)]
```

For the complete version of this more functional approach, see the [solution2_fun.py](#) file.