1)
Write a query to list all projects and the total # of hours dedicated to it.
Use "Project Name" as the first column and "# Hours" as the second column.
Order everything by the "# Hours" column.

```
+-----------------------------------------------+
| projects                                      |
+-----------------------------------------------+
| id BIGINT UNSIGNED PK NOT NULL |
| name VARCHAR(100) NOT NULL       |
+-----------------------------------------------+
```

```
+-----------------------------------------------------------------------------------+
| projects_efforts                                                                  |
+-----------------------------------------------------------------------------------+
| id BIGINT UNSIGNED PK NOT NULL                                                    |
| project_id BIGINT UNSIGNED PK NOT NULL FK(projects.id) |
| hours INT UNSIGNED NOT NULL                                                       |
+-----------------------------------------------------------------------------------+
```

2)
Some people here at DB1 got a sugar rush from drinking too much Tubaína and thought it would be a good idea to build a new OS from scratch, named DB1OS.

Unfortunately, due the lacking of testing and actually thinking the project through, the development team has found two major issues with how memory addresses are handled within its DB1OS's kernel.

Those are:
1)
- If a process `T` should be allocated in a given memory address `x` at `(r, c)`;
- and the `x` address is already being used by another process `R`;
- `x` and all its adjacent addresses used by `R` must be replaced by `T`.

2)
- If a process `T` should be allocated in a given memory address `x` at `(r, c)`;
- and the `x` address is free;
- `x` and all its adjacent addresses also free must be allocated to `T`.

You are in charge to solve these issues by implementing the `allocatePIDAtMemorySlotAddress` function.

The following schematics are in PHP-like, but you can solve it in any language you want as long as you follow the assessment constraints.

SCHEMATICS:

```
function allocatePIDAtMemorySlotAddress(
	string $processPID,
	MemorySlotAddress $memorySlotAddress,
	array $memoryPool // array of arrays where each represents a row and its inner positions the columns of the memory pool matrix
): MemoryPool {}

interface MemorySlotAddress {
	public function getRow(): UnsignedInteger;
	public function getColumn(): UnsignedInteger;
}
```

EXAMPLES:

Example 1:
processPID = D
memorySlotAddress = (2, 4)
memoryPool = [
    [A 0 0 0],
    [A A 0 B],
    [0 0 0 B],
    [C 0 C 0],
]

Expected output: [
    [A 0 0 0],
    [A A 0 D],
    [0 0 0 D],
    [C 0 C 0],
]

Example 2:
processPID = B
memorySlotAddress = (2, 4)
memoryPool = [
    [A 0 0 0],
    [A A 0 B],
    [0 0 0 B],
    [C 0 C 0],
]

Expected output: [
    [A 0 0 0],
    [A A 0 B],
    [0 0 0 B],
    [C 0 C 0],
]

Example 3:

processPID = D
memorySlotAddress = (4, 1)
memoryPool = [
    [A 0 0 0],
    [A A 0 B],
    [0 0 0 B],
    [C 0 C 0],
]

Expected output: [
    [A 0 0 0],
    [A A 0 B],
    [0 0 0 B],
    [D 0 C 0],
]

Example 4:
processPID = D
memorySlotAddress = (1, 3)
memoryPool = [
    [A 0 0 0],
    [A A 0 B],
    [0 0 0 B],
    [C 0 C 0],
]

Expected output: [
    [A D D D],
    [A A D B],
    [D D D B],
    [C D C 0],
]