**Goblin Recursive Function**

My Goblin recursive function is called *bool Goblin::smell(Player\* player, int smellRange);*. The function is located in the *Monster.cpp* file.

**High-level description of Goblin movement function**

The boolean function takes in a pointer to the player, to access its current position, and its smell range, which is determined by the user. First, the function calculates the distance between the player and the Goblin. The first base case is whether the distance is less than or equal to the Goblin’s smell distance, if it is not, immediately return false. The second base case is whether the Goblin is directly adjacent to the player, if so, return true. If not directly adjacent, the recursive step checks all possible paths by calling itself moving 1 step in each direction and decreasing the smell distance. If any of the recursive calls returns true, then the player can be detected within the smell range. If not, the Goblin can not reach the player with its smell range and the function returns false.

**High-level description of levels generated with rooms connected by corridors**

Each level has its own Temple board that is created at runtime. In the Temple constructor, a 70 by 18 grid of ‘#’ is created. In the constructor *generateRooms()* is called which randomly creates between 4 and 7 rooms. For each of these rooms, a random width, height, and top-left coordinate is generated. After each room is generated, it is compared with the already created rooms to ensure they do not overlap. If they do not, then a new room, made of ‘ ‘, is created until they do not overlap. After the rooms are created, the constructor calls *connectRooms().* For each room, the distance formula is used to determine the closest room to itself, then it creates a path to it if a path does not already exist. But, to account for the small probability that not all the rooms are connected, a depth-first search function is called. The *generateRooms()* and *connectRooms()* functions are called until all the rooms are connected. To enter a new level, every time a player is on the stairway, and clicks the appropriate key, a *newLevel()* function regenerates a new board with new monsters and objects, deleting the old board but saving the player’s progress.

**I implemented all of the game functionality.**