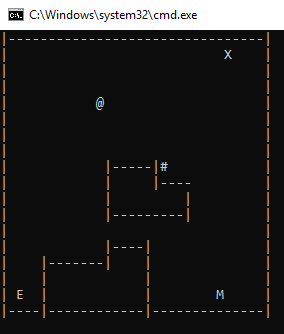
### C# 101 Project Brief

Create a Roguelike RPG game on the console! Here are the requirements:

1. Levels: each **level** is represented by a **map** that has an **entry** and an **exit**.
2. Maps\Levels: each map has **walkable area**, w**alls**, the player's **avatar**, **enemies, traps** and **interactables**. Use the different ASCII characters to build the map graphics to the console. The map’s boundaries must be symmetrical. The player can move about the map and interact with its elements.
3. Input: Movement should be instantaneous - the player presses a button and his avatar moves.
4. Example Map:  
     
   **Where E** is the Entrance  
   **X** is the Exit  
   **|** is a vertical wall  
   **-** is an horizontal wall  
   **@** is the player1  
   **M** is an enemy  
   **#** is a treasure chest  
   **Blank Space** is walkable  
   **You can use your own symbol system.**
5. Entries & Exit: Entries don’t do anything, just show where the player started the level. Exits lead to a new level.
6. Progression: each level is harder to play but is more rewarding.
7. Procedural Maps: the maps in the game are created procedurally, implement an algorithm that will generate the maps at runtime.
8. Data: Print to the player game events and data to the bottom of the map. E.g., “You advanced to maze 3!”, “You found a broad sword which grants you 5 damage”, “The Goblin hits you for 3 damage, you now have 20 hp”. Keep this log section at least 5 entries tall
9. Attributes: Decide on some attributes the player character has. You must have representation for **Damage** and **HP**, but you can create a more complex system if you want (with strength, agility etc.).
10. Interactables:
    1. Traps: traps are invisible to the player until he steps on them.
    2. Monsters: monsters can be fought. Monsters move towards the player once in range
    3. Treasure Chests: treasure chests reward the player (with attributes, money etc.)

**Choose at least four extra credit**

Extra Credit

1. Create an options menu that can set the map generation algorithm parameters, console logs count and other aspects of gameplay
2. Create a HUD to present the player important data such as HP, armor, etc.
3. Create an inventory system
4. Create enemies bigger than one tile
5. Add a save and load system
6. Create a more elaborate AI for enemies
7. Make the maps asymmetrical
8. Make an elaborate combat system that uses damage reduction, evasion, critical hits etc.
9. Use colors within the console to denote different entities
10. Create shops that sell items and currency that can be dropped
11. Add **various** sound effects

### Evaluation

* Core: the core of the game at runtime, per described in the assignment
* Encapsulation: **good** uses of classes and methods to break your code into components.
* Data Structures: use of suitable data structures to represent collections of data in your game
* Code Organization: your code structure, organization (spacing, tabbing, etc.) and naming conventions
* Extra features: any extra features implemented properly, will grant bonus score

### Delivery

* Delivery is set to the 14th unit, March 1st, 2021.
* Deliver your projects unto the appropriate folder on the drive
* Use the “How to Submit Your Homework” for further guidelines