The first goal will be to recreate the UML plan in header files, which will be quite tedious and probably destroy my will. (There are probably automatic tools for this which I'll look for.) This might take an hour or so.

Then the initial step would be to set down enough of the code that the basic floor layout can be displayed (only the walls/doors/floors). This will be done by July 20.

Random generation comes next, with all entities being nothing more than symbols. Given the emphasis on generation order in the document, this probably needs to be heavily tested. Will be done by July 21.

Accepting input correctly and implementing a responsive player character go together. Hopefully the PC will not walk through walls. Will be done by July 23.

Including the items shouldn't be too bad, since they're fairly straightforward. Managing potion memory will probably be the finest point. Will be done by July 24.

Making sure the enemies work correctly will be tricky, especially the merchants and potion sales (which weren't really described in detail). Will be done by July 26.

Allocate a day to test various scenarios.

The rest of the time will be spent trying to implement the DLC and failing.

Question: How could your design your system so that each race could be easily generated? Additionally, how difficult does such a solution make adding additional classes?
Answer: In my planned system, a race is simply a decorator that implements the basic traits of a character (attack, defense, etc.). Any additional races or classes would simply be new decorators in place of or alongside the current ones.

Question: How does your system handle generating different enemies? Is it different from how you generate the player character? Why or why not?
Answer: No; from the point of view of the Character class, there is no distinction between players and enemies. All characters are differentiated by their trait decorators (and of course their current health etc.). Player input is handled outside of the character.

Question: How could you implement special abilities for different enemies? For example, gold stealing for goblins, health regeneration for trolls, health stealing for vampires, etc.
Answer: I would simply modify the racial trait decorator; for example, since attacking is handled by the trait I could simply change the attack(Character target) method so that any damage done heals the attacker.

Question: What design pattern could you use to model the effects of temporary potions (Wound/Boost Atk/Def) so that you do not need to explicitly track which potions the player character has consumed on any particular floor?
Answer: I'm modelling temporary potion effects as "extrinsic traits", which are added to the decorator stack and cleared when the character leaves a floor.

Question: How could you generate items so that the generation of Treasure and Potions reuses as much code as possible? That is, how would you structure your system so that the generation of a potion and then generation of treasure do not duplicate code?
Answer: I'm not really sure how there would be duplicated code in the first place, other than the code to find a random location which can of course be placed in a separate function.