Plan of Attack

# Breakdown of Assignment

# Estimation of Completion

# Role of Group Members

# Project Questions

**Question 1: How could your design your system so that each race could be easily generated? Additionally, how difficult does such a solution make adding additional races?**

**Question 2: How does your system handle generating different enemies? Is it different from how you generate the player character? Why or why not?**

**Question 3: How could you implement the various abilities for the enemy characters? Do you use the same techniques as for the player character races? Explain.**

**Question 4: The Decorator and Strategy patterns are possible candidates to model the effects of potions, so that we do not need to explicitly track which potions the player character has consumed on any particular floor. In your opinion, which pattern would work better? Explain in detail, by discussing the advantages/disadvantages of the two patterns.**

Answer: In our opinion, the strategy pattern works better for this class, as we are planning to add a double dispatch function to the potion and hero class; then change private fields of our ‘hero’ with respect to the effect of the potion. At the beginning of the game, after selection of character race, we record the character’s default properties, and reset some properties (like Atk or Def) when the ‘hero’ enters a new floor. However, we may face some difficulties working on DLCs for potions. For example, if we add a permanent boost atk potion, the effect may be conflict with the reset function at the beginning of each floor.

**Question 5: 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 does not duplicate code?**

Answer: Template method can be used for generating items. Since generating functions of potion and treasure has lots of similarities, we may design them into one template function that handle both items. Thus, we may avoid some duplicate code.