**Ivey Monopoly Overview**

Our plan is to model an interactive game of Monopoly in Excel VBA. The model will allow the user to play a game of monopoly with up to 3 other players by process of taking turns in inputting game decisions. Ivey Monopoly will be modeled closely to the original Monopoly board game; complete with a game board of 28 properties along with chance and community chest card equivalents. The object of the game is to become the wealthiest player as measured by game money through buying, renting and selling of property. Due to the long duration required to complete the average Monopoly game, there will be a time setting that declares a winner according to game money, once the selected amount of time has expired.

**Functionalities**

As illustrated by this figure, the excel model will have a game board consisting of 28 different properties, 2 Chance, 2 Community Chest, Go To Jail, Jail, Go and Free Parking spaces. The simulation of rolling dice will be provided and the model will automatically move the player’s piece around the board according to the randomly generated number of steps provided by the “dice”. Depending on where the player lands, different actions will be prompted for the players to either make decisions that requires inputting data or selecting among the given options. For example, if the player lands on an unsold property, they will be prompted to select between the option of purchasing or declining to purchase. If the player lands on a property owned by another player, they will be prompted to pay the associated property rent. If the player lands on a Chance or Community Chest card, they will be subject to a random event that may take the form of a fine or reward to add to their game money. Special board locations where the player can land include “Go To Jail” and the actual “Jail” section. “Go To Jail” will immediately send the player’s game piece to “Jail”. Within “Jail” the player can only move their piece out if they attain rolling doubles in their dice simulation or via a fine or “Get Out of Jail Free Card”. After the associated inputs are made by the player after rolling dice, the model will end the players turn and loop back to the procedure of rolling dice for the next player. The nature of monopoly requires the model to provide players the ability to keep track and manage their properties. This property ownership functionality is essential to game play aspects such as building houses to increase rent, mortgaging and selling properties. The model will provide the player a similar kind of property ownership as accomplished by property deed cards in the real Monopoly board game. Another essential component of the Ivey Monopoly model will be to keep track of the different amounts of game money owned by each player and the rules governing how this game money is increased and decreased through collecting/paying rent property, buying/selling properties or financial rewards/fines from Chance and Community cards. The rules governing how game money dictates bankruptcy such as when the player has 0 game money will also need to be provided.

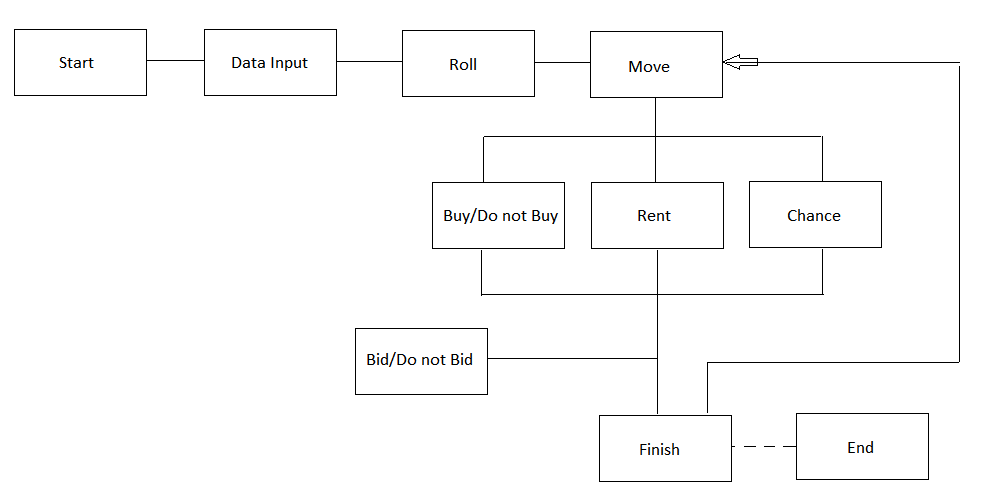
**Inputs and Outputs**

Inputs required from player include the names of the players, the selection of the numbers of game players, the time limit of the game, whether to buy a property or not upon landing, whether to pay fine to get out of jail, offers to buy property from other players and bidding price

Outputs from the model include a simulated dice roll, randomized chance/community chest outcomes, total game money indicator, comprehensive list of properties acquired, winner and bankruptcies based on the amount of game money held by each player.

**Design**

After careful analysis and discussion we were able to conceptualize the process of playing the monopoly game from start-to-finish. The following work flow is a basic map of how the user experiences the program and we will assign functionalities to each box.



Once all functionalities are created, they will be converted to programming classes. Once we have a clear idea of what the classes are we will be able to create another conceptual framework linking all the classes together. Each class will be constructed with the three following sections: Input, Process and Output. For example, for the class START TURN (the user begins to move), the input would be the player, and his/her current location. The process would be to call the ROLL DICE class, call the PROPERTY INFO class and move the player. The output will not return anything. We believe this process will be more organized and ensure that cross-compatibility between different programmers.

**Code Generation**

Afterwards, the amount of work will be divided among group members and the process of code generation will be initiated. In addition, relevant classes will be split into modules. Once the functionalities have been completed by the programmers, we will begin to develop the graphic user interface. The graphic user interface will be able to call the different classes via buttons, scroll bars, etc. Once the preliminary program has finished, we will add documentation and non-essential features that the Monopoly game has.

**Testing**

For quality assurance purposes, testing will begin at the conceptual level. We will run extreme cases through the concept in the design phase to ensure that the program is robust and all types of user interactions are accounted for. During code generation, each class will be tested individually and again within its own module to ensure the program runs as intended. How we will approach testing will be to use two extreme cases at the end of both spectrums and one case in the middle. This approach will allow for minimal error to occur. Once the program is complete, we will test the program from the perspective of the user through the graphic user interface to ensure that all bugs are smoothed out before final delivery.