How to play game with 2 players:
In command prompt Go to directory
for instance : onid/dominion
type in command prompt : make all
To play game Type in command prompt playdom 2
It plays game with 2 player. You can see the step.

Documentation:
initailizeGame method.

****PutSeed((long))randomSeed), random seed is assigned a random number.

```
if (numPlayers > MAX_PLAYERS || numPlayers < 2)
    {
      return -1;
    }
```

if number of players are less then 2, game cannot be played. If number of players are more then maximum players, we don't have enough cards so game cannot be played. Return -1, invalid.

```
//check selected kingdom cards are different
  for (i = 0; i < 10; i++)
    {
      for (j = 0; j < 10; j++)
        {
          if (j != i && kingdomCards[j] == kingdomCards[i])
            {
              return -1;
            }
        }
    }
```

all the kingdom cards should be different type. Make sure that we don't have two types of smithy or two types of adventurer. There can be multiple smithy cards, but as a type, smithy is unique and same is true with adventurer and other cards.

```
//set number of Curse cards
  if (numPlayers == 2)
    {
      state->supplyCount[curse] = 10;
    }
  else if (numPlayers == 3)
    {
      state->supplyCount[curse] = 20;
    }
  else
    {
      state->supplyCount[curse] = 30;
    }
```

More the players, more curse cards will be part of the game.

```
//set number of Victory cards
  if (numPlayers == 2)
    {
      state->supplyCount[estate] = 8;
      state->supplyCount[duchy] = 8;
      state->supplyCount[province] = 8;
    }
  else
    {
      state->supplyCount[estate] = 12;
      state->supplyCount[duchy] = 12;
      state->supplyCount[province] = 12;
    }
```

Remember, victory cards are duchy, estate and province. More players, more victory cards are needed to make game interesting.

```
//set number of Treasure cards
  state->supplyCount[copper] = 60 - (7 * numPlayers);
  state->supplyCount[silver] = 40;
  state->supplyCount[gold] = 30;
```

supply decks starts with a deck of 30 gold, deck of 40 silver and deck of some copper. Each player gets 7 copper to start with so supply deck for copper has 60 - (7 * numPlayers) cards.

```
//set number of Kingdom cards
  for (i = adventurer; i <= treasure_map; i++)          //loop all cards
    {
      for (j = 0; j < 10; j++)                          //loop chosen cards
          {
            if (kingdomCards[j] == i)
              {
                //check if card is a 'Victory' Kingdom card
                if (kingdomCards[j] == great_hall || kingdomCards[j] == gardens)
                      {
                        if (numPlayers == 2){
                          state->supplyCount[i] = 8;
                        }
                        else{ state->supplyCount[i] = 12; }
                      }
                else
                      {
                        state->supplyCount[i] = 10;
                      }
                break;
              }
            else    //card is not in the set choosen for the game
              {
                state->supplyCount[i] = -1;
              }
          }

    }
```
great_hall and garden cards can have varying number depending upon number of players. If number of players are 2, we will have 8 great_hall and 8 gardens to start with, else we will have 12 gardens and 12 great_hall to start with. For all other cards, we start with 10 cards. So, for smithy, adventurer, ... we start with 10 cards.

```
for (i = 0; i < numPlayers; i++)
    {
      state->deckCount[i] = 0;
      for (j = 0; j < 3; j++)
        {
          state->deck[i][j] = estate;
          state->deckCount[i]++;
        }
      for (j = 3; j < 10; j++)
        {
          state->deck[i][j] = copper;
          state->deckCount[i]++;
        }
    }
```
Each player start with 3 estate and 7 copper cards. Total 10 cards.

```
//shuffle player decks
  for (i = 0; i < numPlayers; i++)
    {
      if ( shuffle(i, state) < 0 )
        {
          return -1;
        }
    }
```
Shuffle deck for all the players.

```
//draw player hands
  for (i = 0; i < numPlayers; i++)
    {
      //initialize hand size to zero
      state->handCount[i] = 0;
      state->discardCount[i] = 0;
      //draw 5 cards
      // for (j = 0; j < 5; j++)
      //   {
      //     drawCard(i, state);
      //   }
```

It doesn't actually draw the cards. Comment is written wrong. Typical mistake in real world programs where someone wrote the comment. Someone else changed the code after few months and forgot to change comment. It just initializes all players hand to 0 and all players discard pile to 0.

```
//Moved draw cards to here, only drawing at the start of a turn
  for (it = 0; it < 5; it++){
    drawCard(state->whoseTurn, state);
  }
```

Cards are drawn here.

drawCard method:

If deck is empty, discard pile becomes the new deck. Shuffle once discard pile is moved to deck. Draw first card from the deck into player's hand.

If deck is not empty, get the first card from the deck into hand.

In dominion.h

```
struct gameState {
  int numPlayers; //number of players
  int supplyCount[treasure_map+1];  //this is the amount of a specific type of card given a specific number.
  int embargoTokens[treasure_map+1];
  int outpostPlayed;
  int outpostTurn;
  int whoseTurn;
  int phase;
  int numActions; /* Starts at 1 each turn */
  int coins; /* Use as you see fit! */
  int numBuys; /* Starts at 1 each turn */
  int hand[MAX_PLAYERS][MAX_HAND];
  int handCount[MAX_PLAYERS];
  int deck[MAX_PLAYERS][MAX_DECK];
  int deckCount[MAX_PLAYERS];
  int discard[MAX_PLAYERS][MAX_DECK];
  int discardCount[MAX_PLAYERS];
  int playedCards[MAX_DECK];
  int playedCardCount;
};
```

Each player has one hand, one deck, one discard pile. Player's hand can start with 5 but it can go upto MAX_HAND. State of the game keeps changing with every steps taken by any of the player. It is most important entity in testing. Any testing should test s**tate of game as per execution of code** with expected game state.