```
Include
// Include library files according to some path environment variable.
{
        "stdlib/stdlib.cgl";
}

CardEntities
// Card Entities are addressed with '$'
// All cards are initially located in the first Card Entity, in normal order.
{
        dealer;
        player0;
        player1;
        flop;
}

Globals // Global variables are addressed with '#'
{
        var currentPot;
        var lastBid;
        var chips;
        var players;
        var doneBidding;
        var message;
}

Start // Deal cards, set chips
{
        var i;
        var j;
        var c;
        <<"Hello World";
        //
        // initialize global variables
        //
        #currentPot = 0;
        #lastBid = 0;
        #chips = [100, 100]; // start players off with 100 chips
        #players = [$player0, $player1]; // have an array of players
        #doneBidding = [true, true];
        #message = "Please select a card.";

        // test card and randomness
        c = H2;
        c = C10;
        c = SA;
        c = D2;
        i = ~1;
        i = ~(5 + 4 / 3);

        // shuffle the deck
        //shuffle($dealer);

        // deal out 5 cards to each player
        //for (i = 0; i < 5; i++)       {
        //   for (j = 0; j < listLength(#players); j++) {
        //       #players[j] <- $dealer[0];
        //   }
        //}
}

Play
// Play functionality associated with a "round" of play
{
        var i;

        <<"Calling Play Function";
        //for (i = 0; i < size(#players); i++) {
        //    play(#players[i]);
```

```
        //}
        i = 0;
        while (i<5){
                i = i+1;
                << i;
        }

           play($dealer);
        while (#doneBidding[0] && #doneBidding[1]) {
           play($player0);
           play($player1);
        }
        //evaluateHandWinner();
}

WinCondition
// Condition for the game to end - evaluated after each round.
// Must return a list of Card Entities (or null if no winner yet).
{
        <<"Checking Win Condition";
        if (#chips[0] <= 0 && #chips[1] <= 0) {
           return [];
        } else {
           if (#chips[0] <= 0) {
              return [$player1];
           } else {
              return [$player0];
           }
        }

        return null;
}

play(var e)
// Play function
{
        var i;
        var bid;

        bid =5;
        <<"Player going: ";
        <<e;
        //i = indexOf(#players, e);
        #doneBidding[0] = false;
        #doneBidding[1] = false;
        //<< "" ^ e; // print the cards of this card entity
        i =0;
        while (i<5)
        {
                i=i+1;
                <<e[i];
        }
        if ( bid == null) {
        <<"bid ok";
        }
        <<"Bid: " ^ bid;
        <<returnFive();
        bid < #lastBid;
        <<"bid 2 ok";
        >>i;
        <<"You gave input " ^ i;

        if (bid == null || bid < #lastBid) // haven't bet or overbet
        {
           >> bid; // input a bet (auto convert input to type of 'bid')
           #chips[0] -= bid; // subtract bet from your current chips
           #currentPot += bid; // add bet to the current pot
           #lastBid = bid; // record the last bid to check overbets
        }
```

```
        #doneBidding[0] = true;

        return;
}

returnFive ()
{
return 5;
}
evaluateHandWinner(var lc, var round)
{
        //The best poker hand wins

        return @1;
}

testfunc()
{

return 0;
}
```