Logic/Psuedo Code

a. Set up arrays that have two elements: Departure, Destination

b. Load the input deck into the array

c. After the query is loaded determine the types of question:

- jumps question

- possible question

- loop question

J*umps* question: *inputs: QueryDeparture (X) and number of jumps (Y)*

1. Prompt the user for a departure city (X)

2. Prompt the user for a destination city (Y)

3. Traverse the departure elements until to find the matching QueryDeparture (X),

(loop the following for the number of jumps)

2. Traverse all of the destination elements where the departure elements = X,  this is 1 jump, traverse the array and report the destination

3. For each of the destination elements, traverse the array where it is the departure element and this is 2 jumps, traverse the array and report the matches

4  Repeat again for each of the destinations that were found in the previous step( this is 3 jumps... and so on and so on)

Can someone get from (X) to (Y)?: *input QueryDeparture (X)*

1. Create scanner input (ask question) departure (X)

2. Create scanner input (ask question) departure (Y)

3. Traverse the departure elements to find the matching query departure point

4. Traverse all of the destination elements where the departure element is the same as the query (scanner input of X) - if the query destination is found toggle stop and report yes,

3. If it is not found, traverse the destination elements again and for each find in the array where that destination is a departure, then traverse that array's destinations  if the query destination is found toggle stop and report yes...

4. Continue until all of the destination element's arrays have been traversed, if you reach the end of the destination elements - report no

L*oop* question:  *input: QueryDeparture (X)*

1. Traverse the departure elements to find the matching the query departure (X)

2. Traverse the departure elements to find the all of the next possible stops  (these destination are one stop away)

3. Traverse all of the destination elements for each of the next possible stops' array and note if X is in the array and if  it aren't only one stop away - round trip available = yes,   if X is not in the array of destinations continue to the next element's array until you find X

4. If it is not found, traverse the destination elements again and for each find in the array where that destination is a departure, then traverse that elements destinations  if the query destination is found toggle stop and report yes...

5. Continue until all of the destination elements have been traversed, if you reach the end of the destination elements - report no