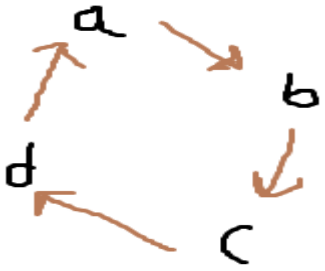
Approach

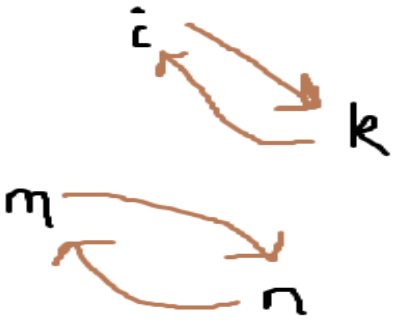
EXPLAINATION IS BIG CODE IS SMALL

- Two or more strings can form a chain if the ending character of the first string is starting character of other string and process followed on all the array elements are connected.
- Now while connecting them at the end we have to ensure that the ending character of the last element of the chain is the first character of the initial element of the chain to complete the Circle
- Now this a graphs question , there is no other way !
- You have to make directed edges from starting position of each string to last character of each string
- we will draw a graph using the first and last character of every string and connected them in order of occurrence
- So vertices will be the first and last character of strings, and we will draw an edge between two vertices if they are the first and last character of the same string, so a number of edges in the graph will be same as the number of strings in the array.
  Now consider { "abb", "bcc", "cxxd" ,"daa" }



  O/p: TRUE

- This graph forms a **eulerian circuit** i.e we any vertex from any vertex
- If there is an eulerian circuit, then chain can be formed, otherwise not.
- **Note: directed graph has eulerian circuit only if in degree and out degree of every vertex is same, and all non-zero degree vertices form a single strongly connected component.**
- the above graph had equal indegree and outdegree and is strongly connected
  Consider some more examples :
  "ikk", "kii" ,"mnn" , "njmmm"



- the above graph has indegree=outdegree for every vertex but 2 components
- O/P: FALSE
- Here indegree!=outdegree
- So first we make sure indegree=outdegree then we check for strongly connected components using dfs and check if any vertex was unvisited
- If we travserse dfs from "a" we visit all vertex and get output as false which is wrong so that's why we check for indegree=outdegree first , then only do dfs once from the **starting position of the 1st string**
  Eg

  Some points to keep in mind

- no of vertex can max be 26 since we have 26 alphabets
- No of edges = no of strings
- First check for indegree-outdegree condition and then check for conencted components
- For checking the connectivity we will implement DFS in the graph if all the vertices are visited then we will return true otherwise false.
- For checking the In and Out degree we'll simply maintain a vector for them and check if both values are equal.

```
for(int i = 0; i < 26; i++){
   if(inDegree[i] != outDegree[i]){
      return false;
   }
}
```

- We'll declare an array of Vector to store edges, this will represent a graph, and maintain another array that will record which character from all the alphabets has to be used and mark them as true