1. I used javascript as my primary language and php for getting contents from a url
2. GetInfo.php returns the contents from a url
3. Under the "Crawler" class, we have **Crawler.Eval.evaluate** function which parses the pattern

4. Implementation example:-

    - 33458
    - *["29160", "48111", "84161"]*
    - 29160
    - *["3382639", "1522687", "8019169"]*
    - HiT the DEADEND:- twice
    - 8019169
    - in the above example, the starting integer is 33458 and the integers converted from the pattern are 29160, 48111, 84161
    - then the 1st integer is accessed and its has 3 patterns which yields again 3 numbers:- 3382639, 15226867, 8019169
    - Now when 3382639 is accessed, it hits a deadend and then it checks for 1522687 which again hits deadend, then 8019169 is accessed
    - At this point:-
        i. listOfNodes =
        {
        "33458":["29160","48111","84161"],
        "29160":["3382639","1522687","8019169"]
        }

        ii. nodeMap = ["33458", "29160", "8019169"]

        iii. visitedNodes = ["33458", "29160", "3382639", "1522687", "8019169"]

5. When all nodes yield "DEADEND" situation:-

    - 81306
    - *["81764", "40958"]*
    - 81764
    - *["304303"]*
    - 304303
    - *["119789", "2013391", "1099020"]*

- HiT the DEADEND:- thrice
- 40958
- *["86089", "74764"]*
- In the above example, after a few iterations, all the integers under "304303" yield DEADEND, then it moves up and checks 40958 because every other integer has already been accessed

6. I added a throw statement when the count goes more than 300 and at that point the crawling stops. Also added console statements to go over the integer values accessed.