**Professor: Amotz Bar-Noy**

**Course: Analysis of Algorithms**

**College: Brooklyn College (City University of New York)**

**Name: Kishan Shah**

**Id: 23970203**

**Realizing Degree Sequences**

**Objective:** Let D be a non-increasing sequence of n non-negative integers in which every d(i) <= n-1. Code a program that decides if D is graphic.

**Method:** Implement the Havel Hakimi algorithm.

**Input:** A sequence of n+1 integers, where first integer is length of the sequence and rest of it is the sequence itself.

**Variants:** HH algorithm has some variants.

**Case-1: Max-HH:**

* The pivot vertex is one of the vertices with the highest degree.

**Case-2: MIN-HH:**

* The pivot vertex is one of the vertices with the lowest degree.

**Case-3: UR-HH:**

* Every vertex is selected as the pivot vertex with a uniform probability.

**Case-4: PR-HH:**

* Vertex v(i) whose current degree is d(i) is selected as the pivot vertex with probability,
* p(i) = d(i) / sum of all (vertices’ degrees).

**Case-5: ParR-HH:**

* Given a parameter x, vertex vi whose current degree di is selected as the pivot vertex with probability,
* p(i) = d(i)^x / sum of all (vertices’ degrees’ power x).

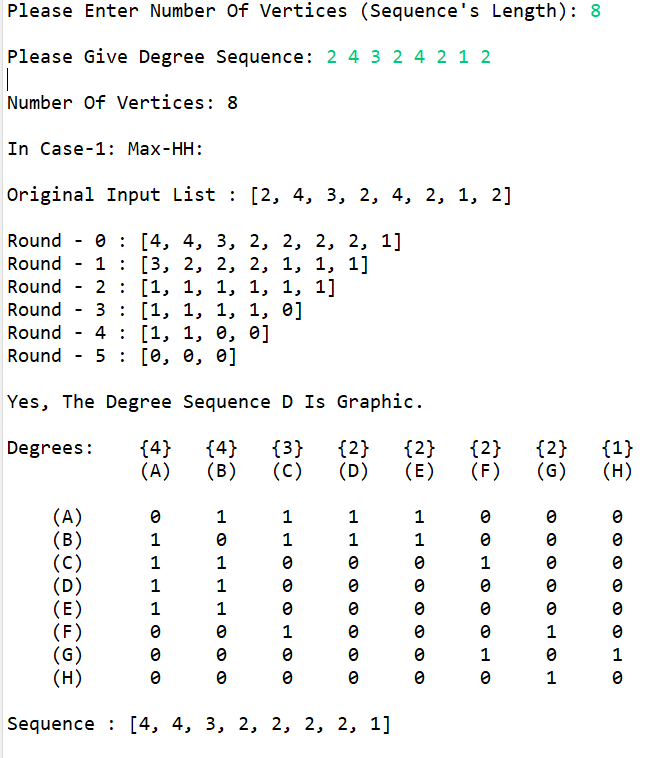
**Observation:**

* As per above cases, we can see that case-1, case-4, and case-5 will generate same output.
* Because in Max-HH, we take maximum degree vertex as a pivot.
* And in PR-HH and ParR-HH, divisor is constant.
* So, basically, whichever vertex has highest degree, will generate higher probability in both the cases 4 and 5.

**My Outputs:**

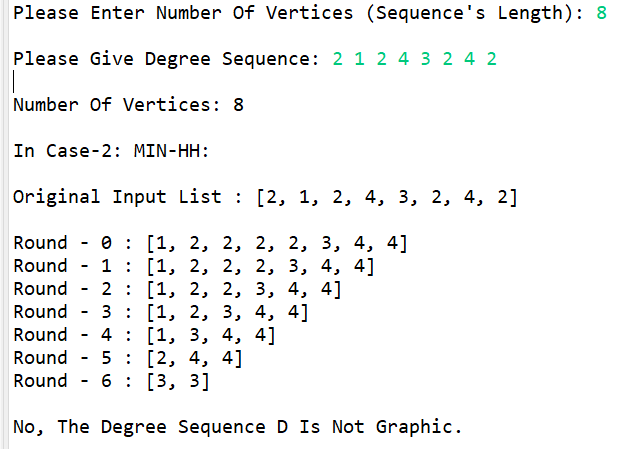
* I am intentionally giving unordered user inputs.
* Different cases will automatically set the sequence list according to their rules.
* So, ultimately, in every scenario, round-0 will represent the original input for the algorithm.
* All test cases are after these sample outputs.

**Case-1: Max-HH:**

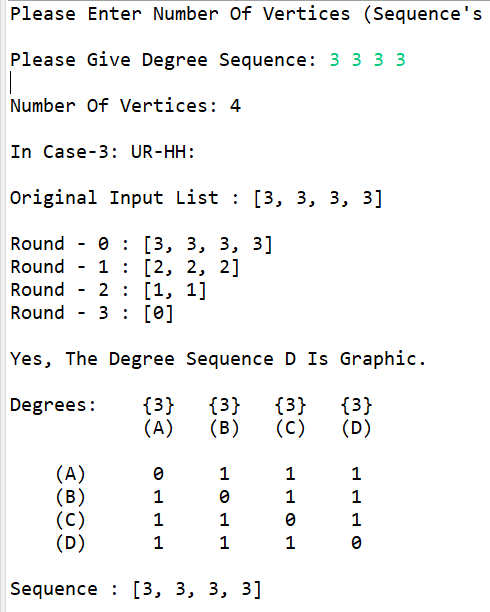


* We can clearly see that if we set the highest degree vertex as the pivot, algorithm works.
* For the case-4 and case-5 as well, input degree sequence will remain same as described in round-0 in above picture.

**Case-2: MIN-HH:**

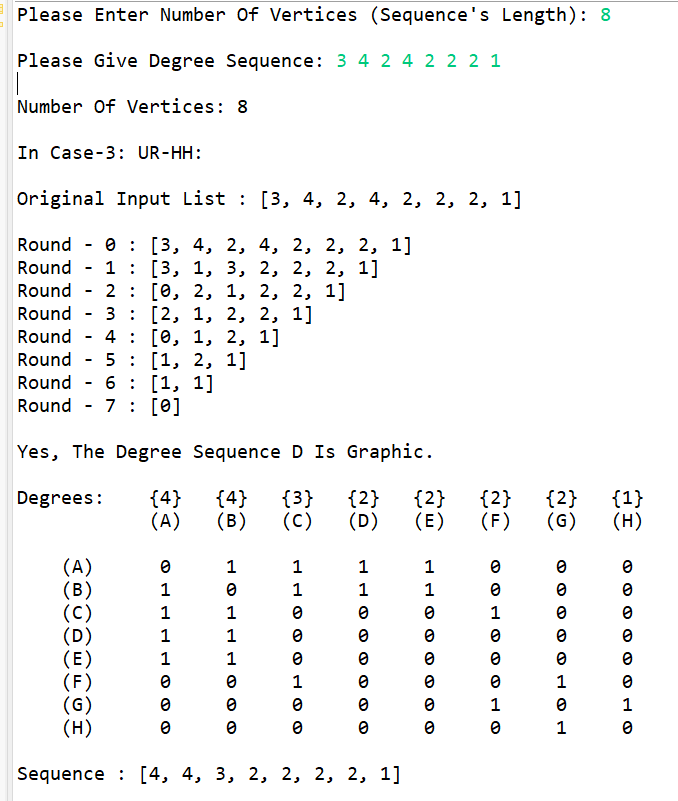


* As we can see that if we choose minimum degree vertex as pivot in above example then algorithm will not work properly.
* But if all degrees are same (complete graph), then even we choose minimum degree vertex as pivot, algorithm will work.

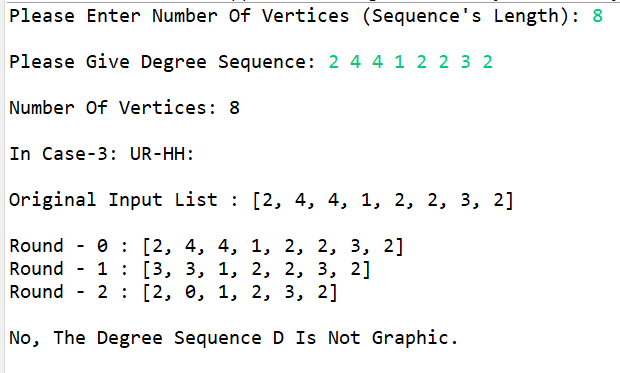


**Case-3: UR-HH:**

* Now, this case totally depends on the input sequence because every vertex has same unique probability.



* In the above example, algorithm worked because of the structure of the algorithm.
* Algorithm first delete first element of the sequence and then subtract ‘1’ from next ‘k’ number of vertices, where k is degree of that deleted vertex.
* But this is not the case in every scenario.
* If we give input like below picture, then algorithm will not work.

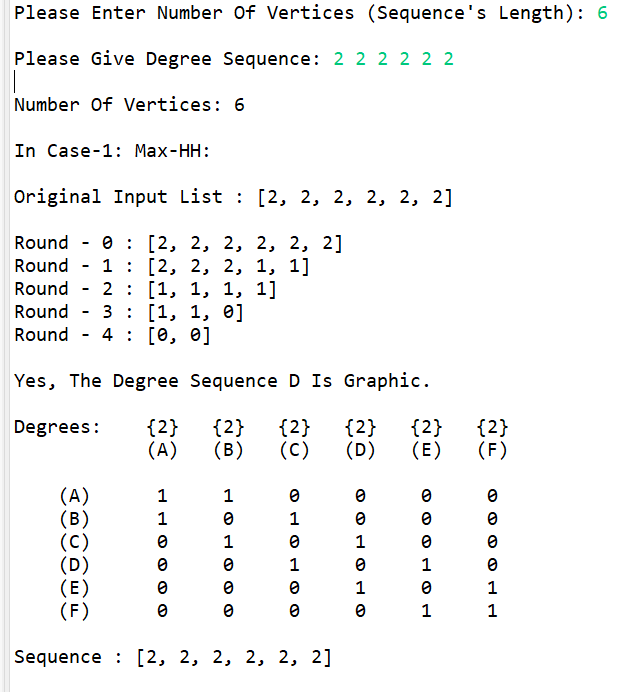


* Hence, for Havel Hakimi algorithm to work properly, we need to give a non-increasing degree sequence as an input.

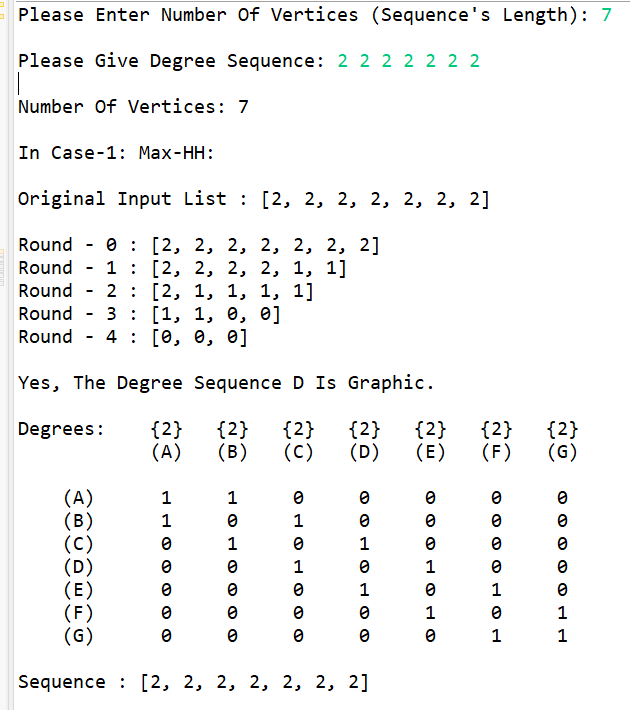
**Testing:**

**Question-1:**

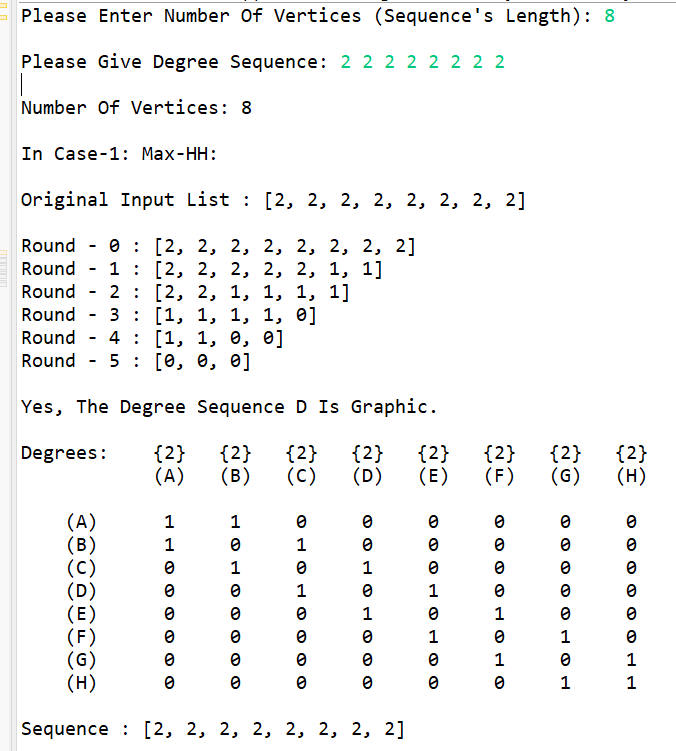
* n = 6.



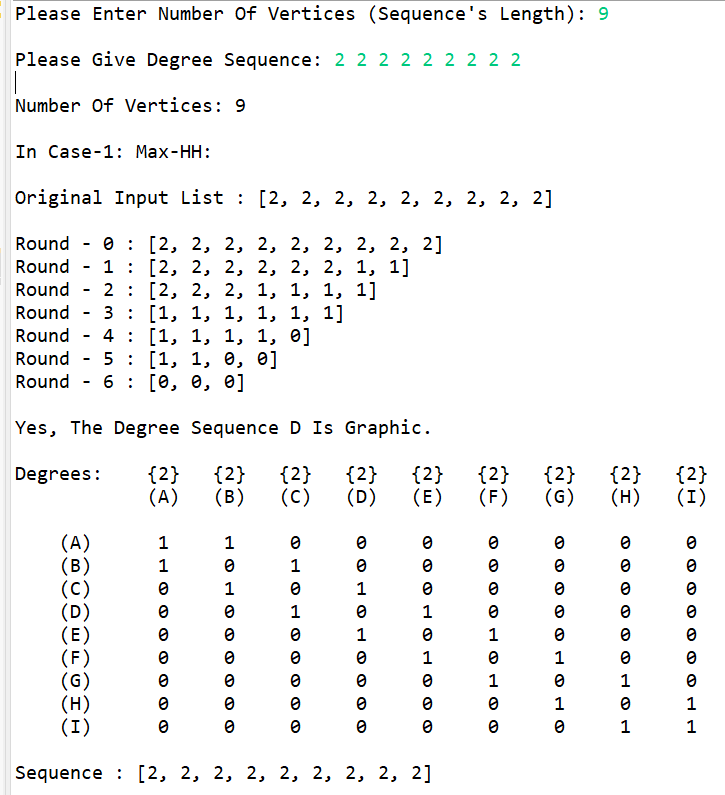
* n = 7.



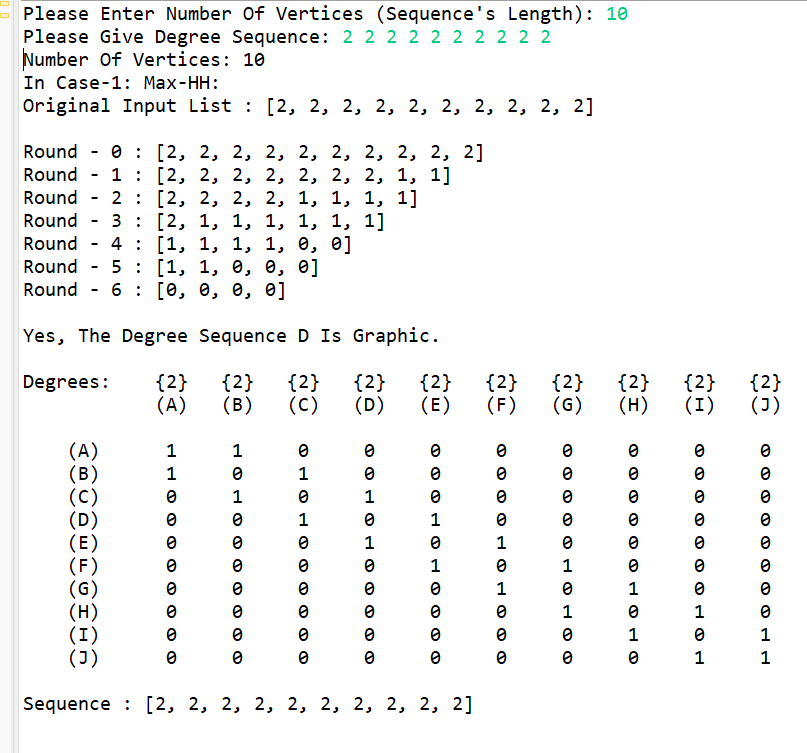
* n = 8.



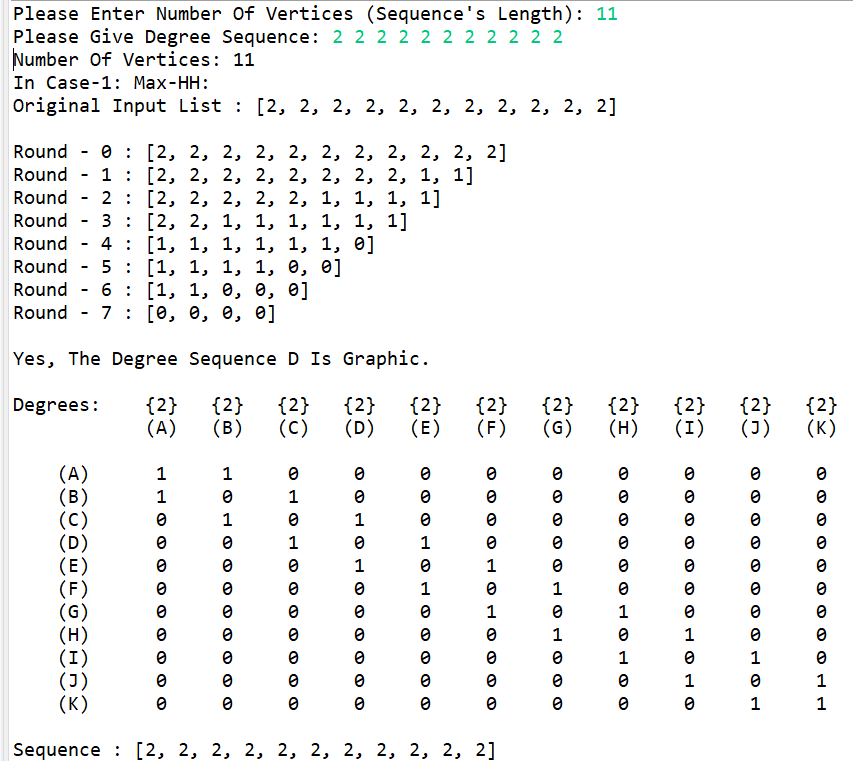
* n = 9.



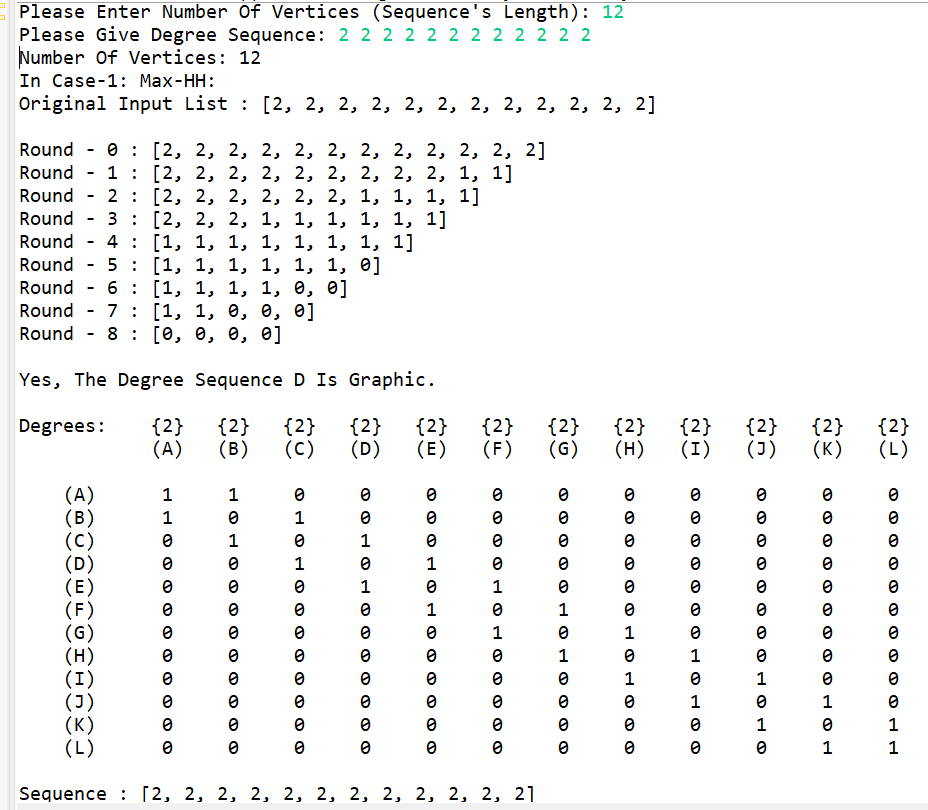
* n = 10.



* n = 11.



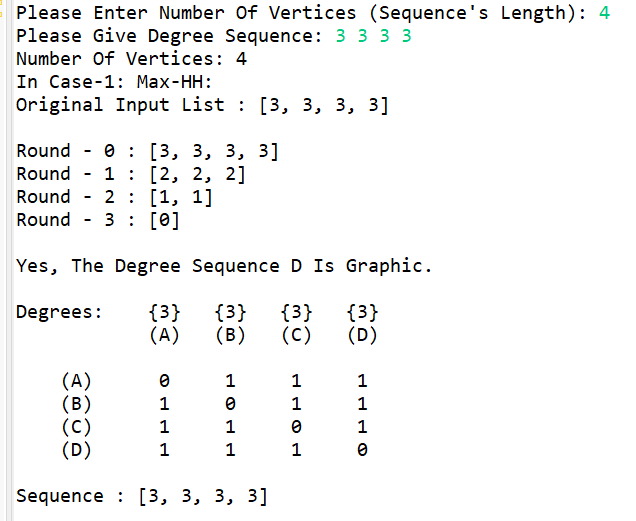
* n = 12.



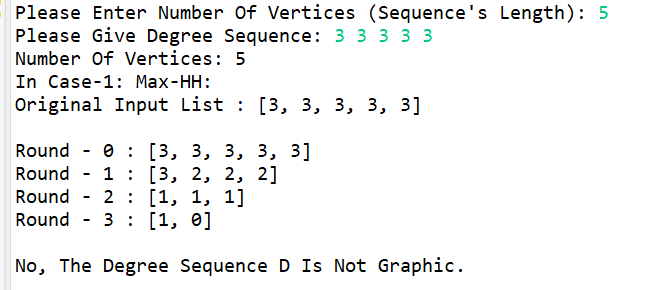
* So, in general, we can say that for any n >= 3, any 2-regular graph is possible.
* For example, graphs having only cycles.

**Question-2:**

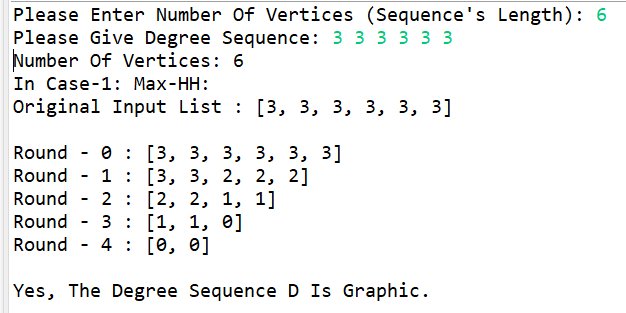
* n = 4.



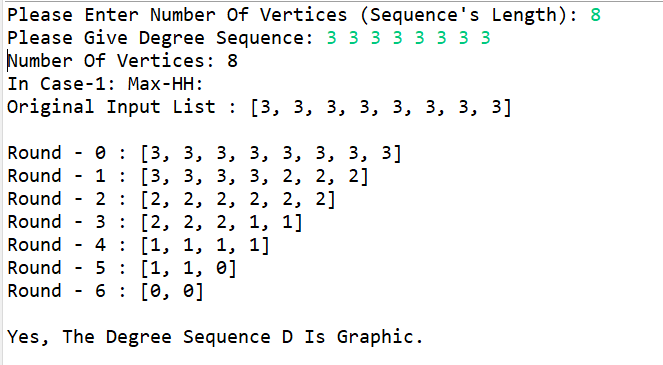
* n = 5.



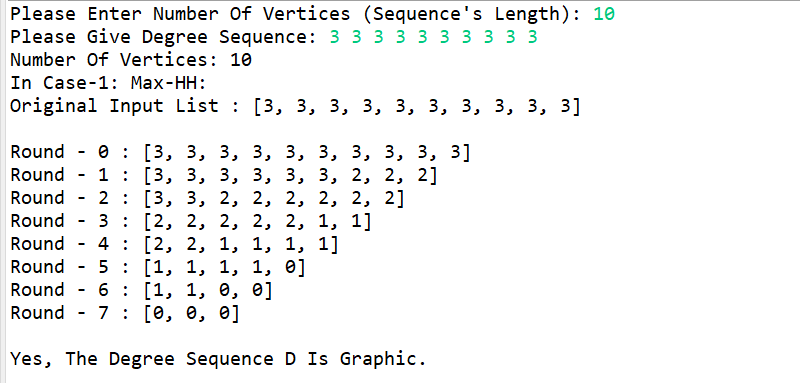
* n = 6.



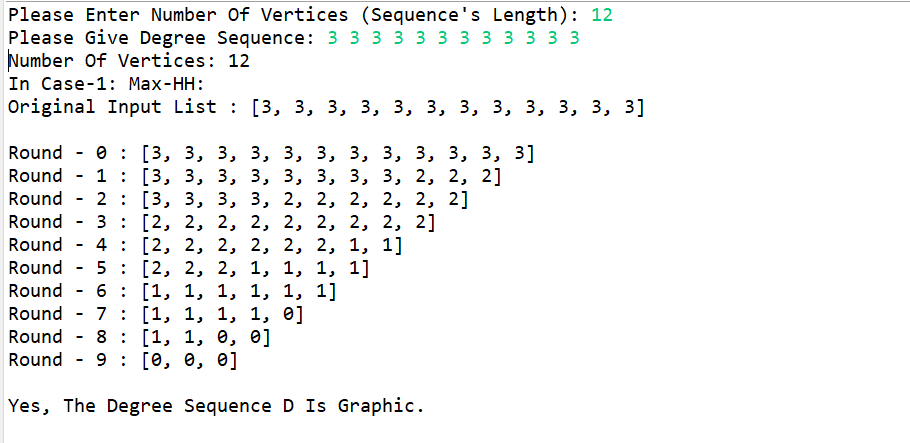
* n = 8.



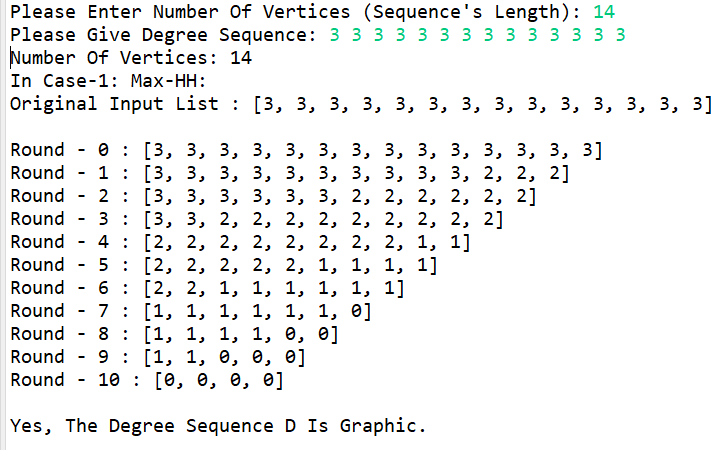
* n = 10.



* n = 12.



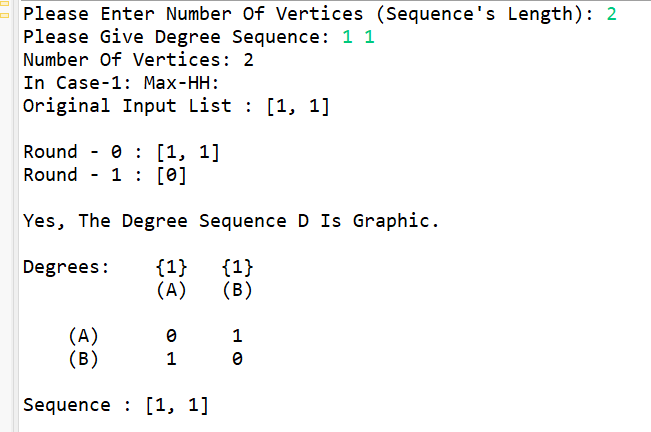
* n = 14.



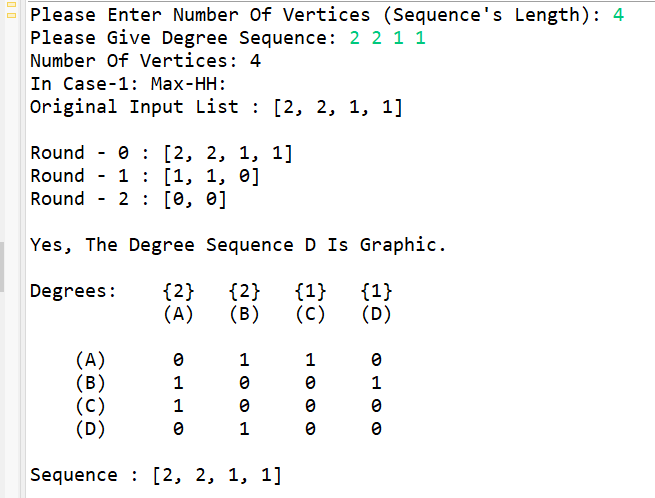
* So, in general, for 3-regular graph, for n >= 4, if n is even then graph is possible and if n is odd then graph can’t be possible.
* Because odd number of n will make total number of degrees odd and which is not possible.

**Question-3:**

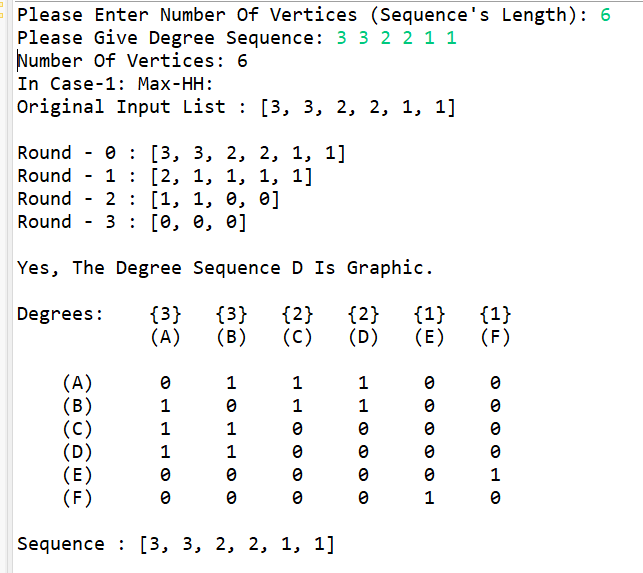
* n = 2.

****

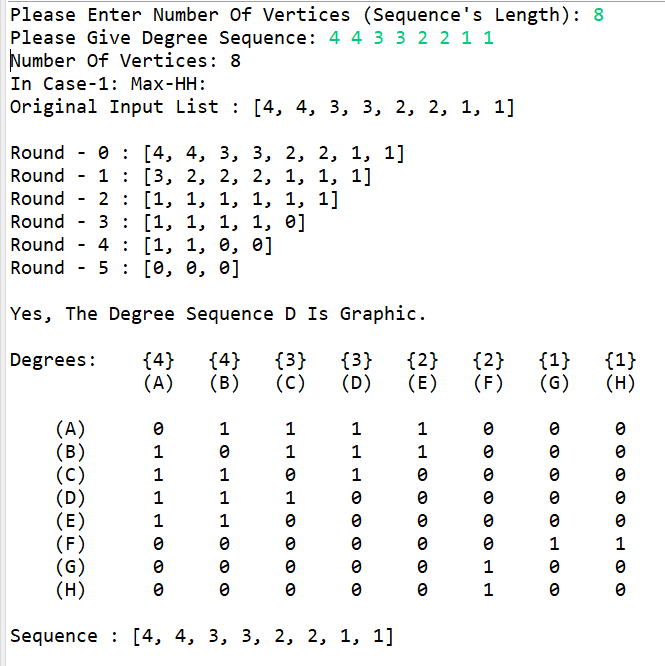
* n = 4.



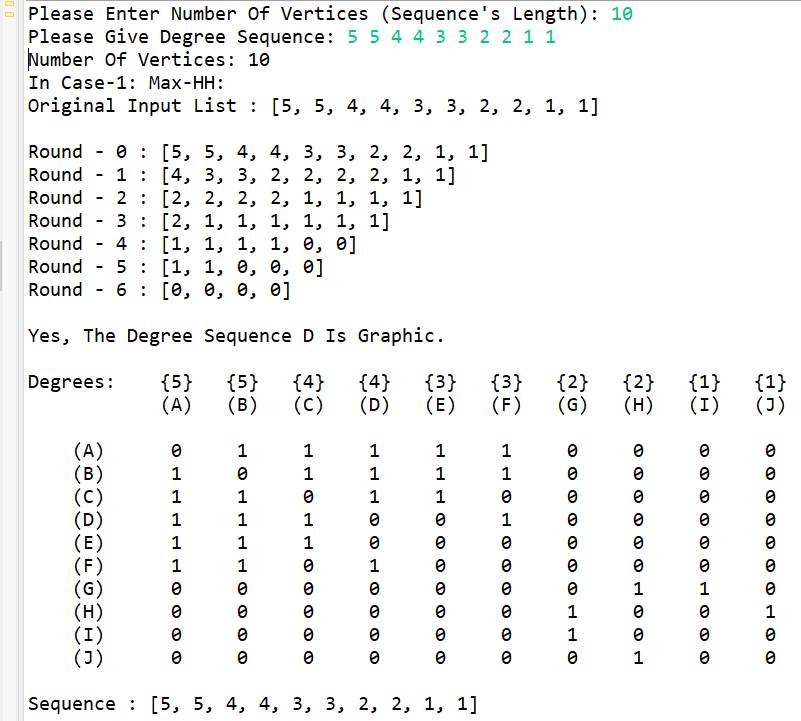
* n = 6.



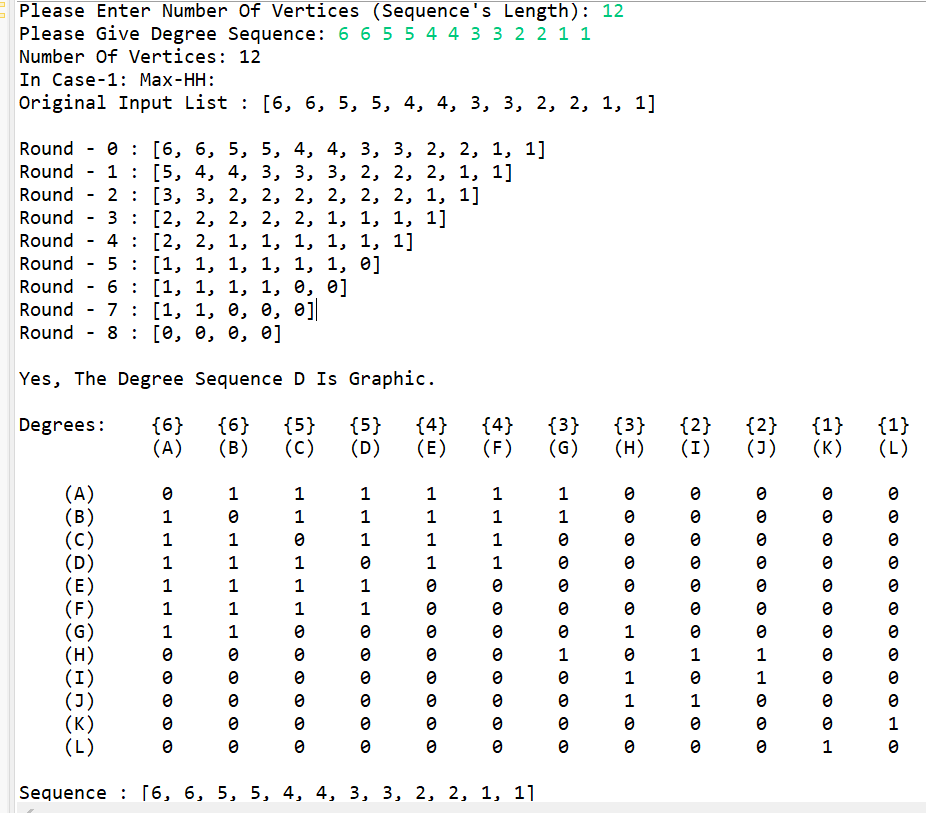
* n = 8.



* n = 10.



* n = 12.

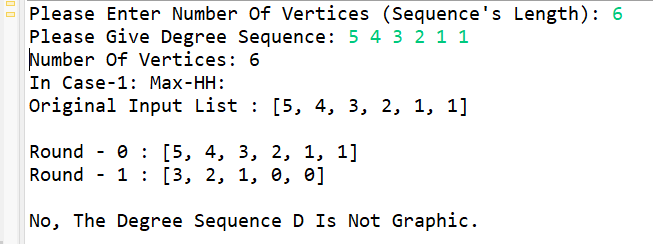


* So, in general, we can say that for even n, the graph is possible, without running the algorithm.

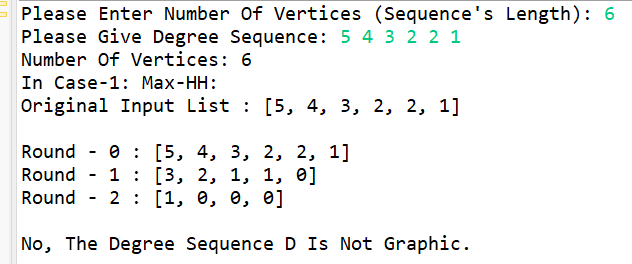
**Question-4:**

(a)

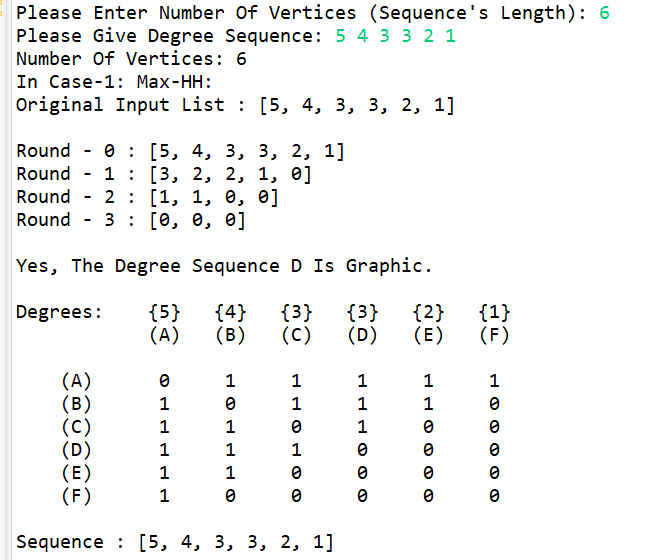
* n = 6, i = 1.



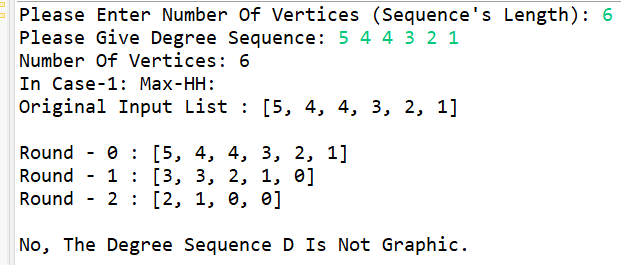
* n = 6, i = 2.



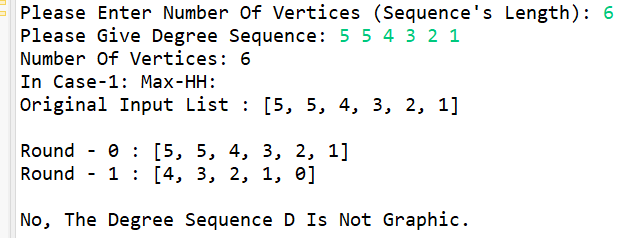
* n = 6, i = 3.



* n = 6, i = 4.

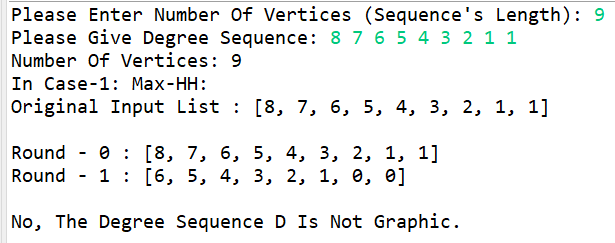


* n = 6, i = 5.

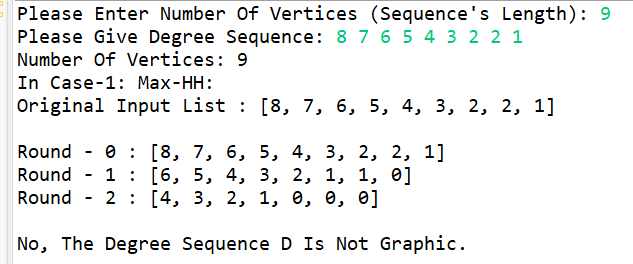


(d)

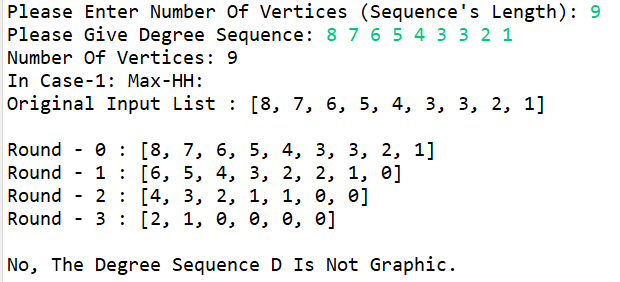
* n = 9, i = 1.



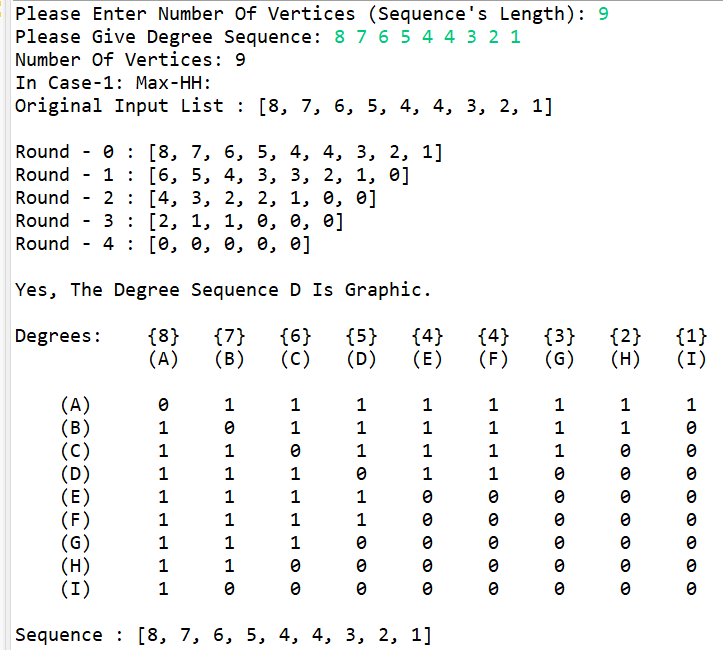
* n = 9, i = 2.



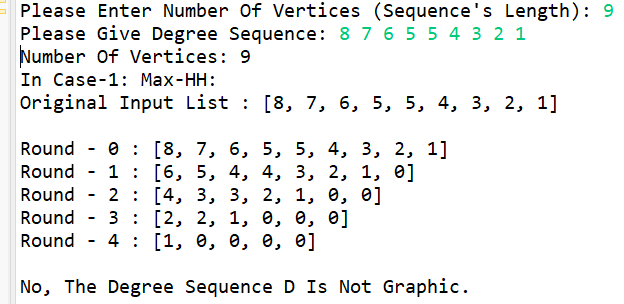
* n = 9, i = 3.



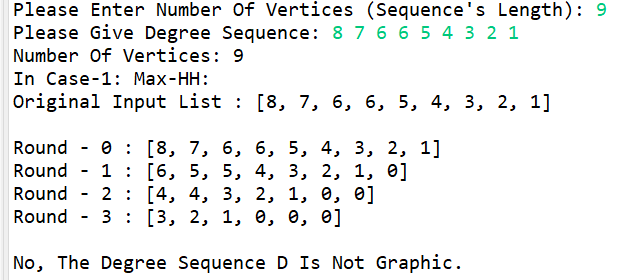
* n = 9, i = 4.



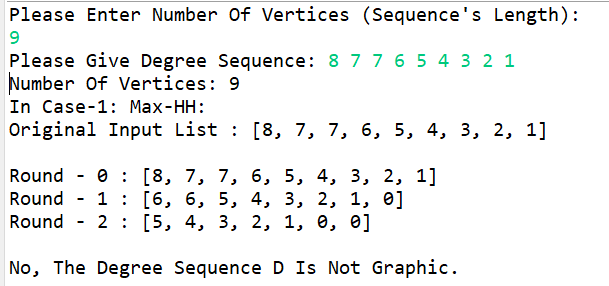
* n = 9, i = 5.



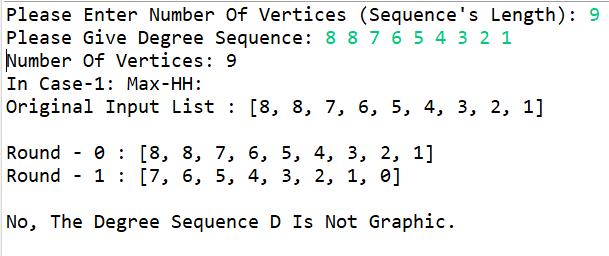
* n = 9, i = 6.



* n = 9, i = 7.



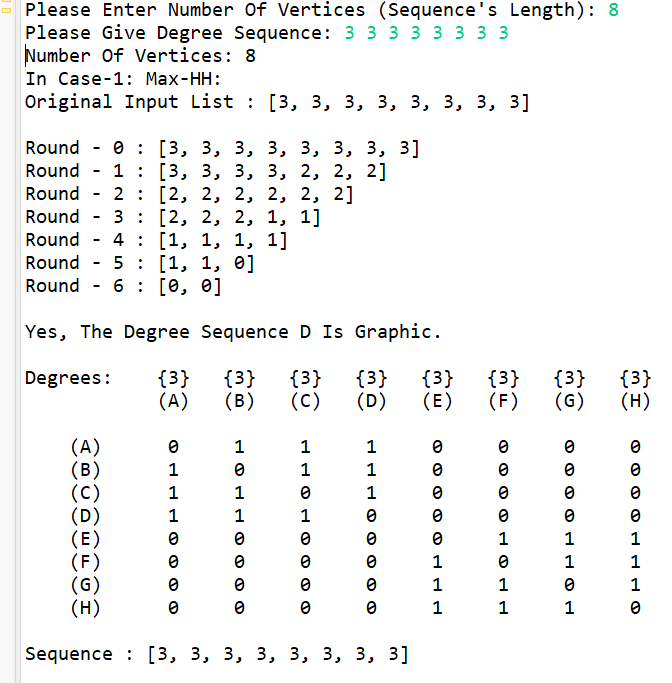
* n = 9, i = 8.



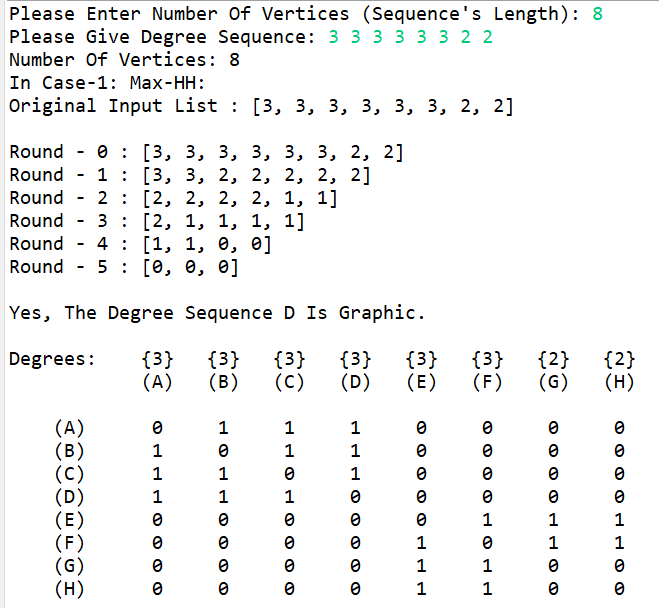
* So, in general, in this kind of structure, when i = floor of(n/2), we will have a graph.

**Question-5:**

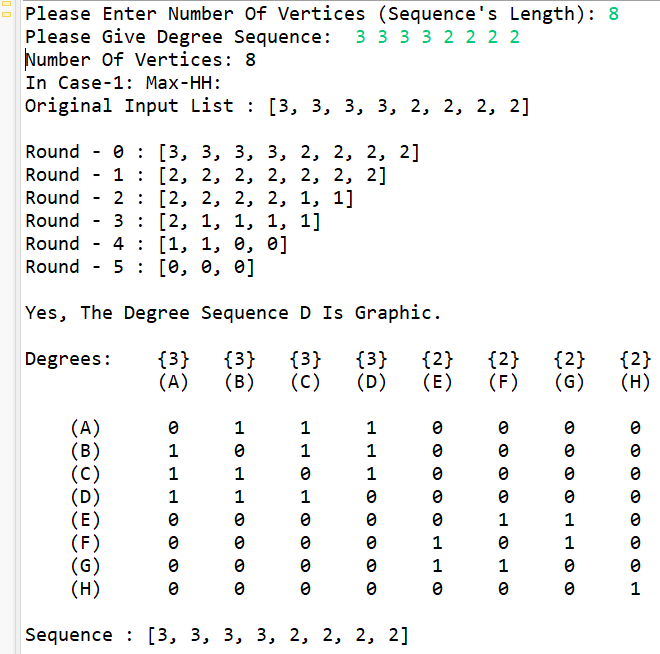
(a)



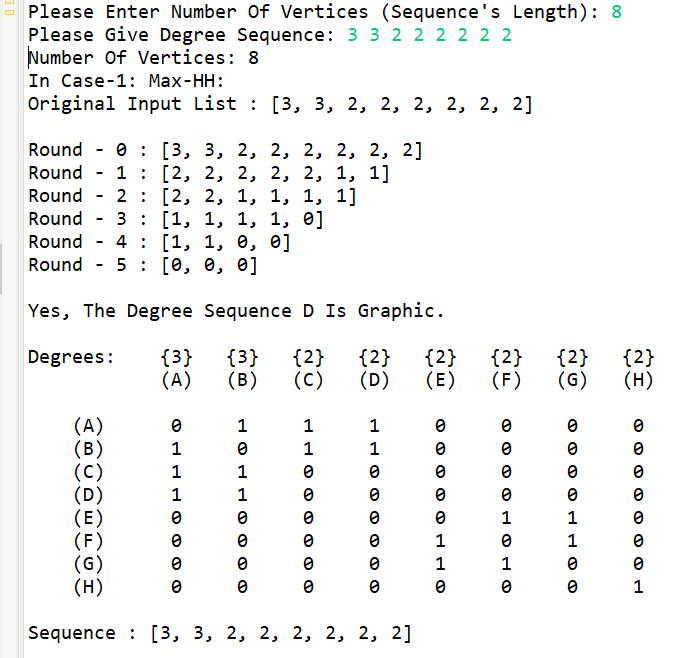
(b)



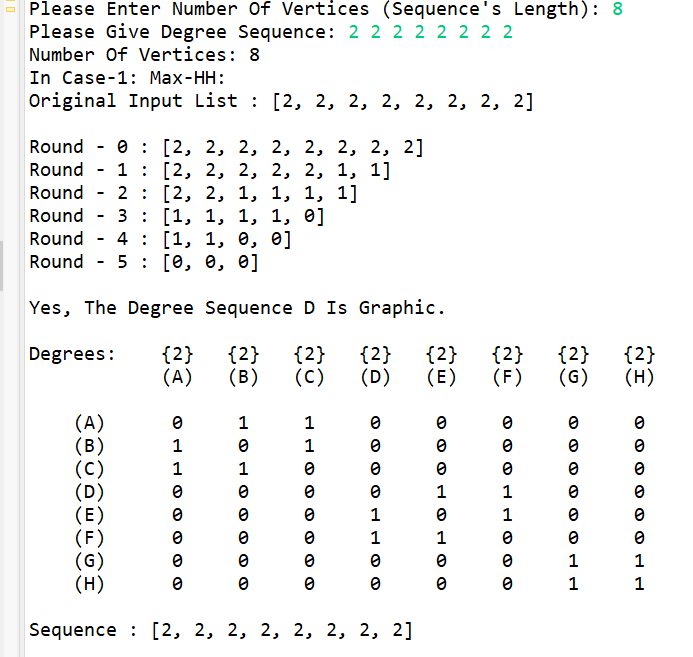
(c)



(d)

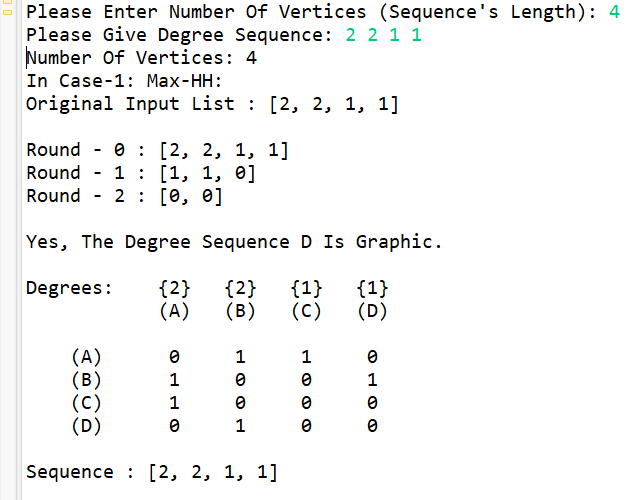


(e)

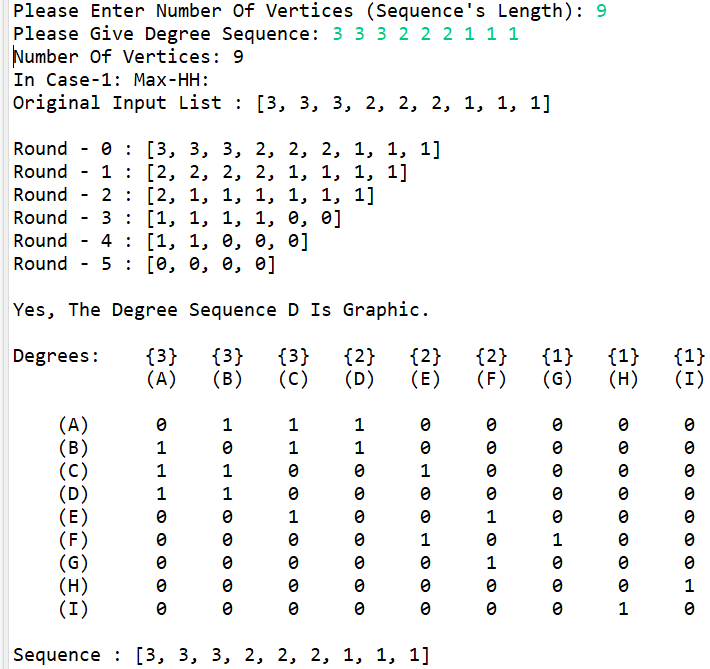


**Question-6:**

(a)



(b)



(c)

