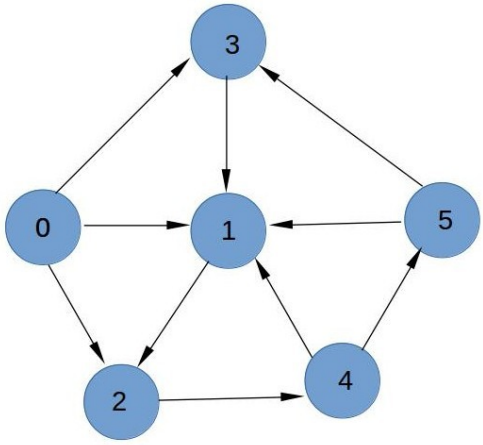
IT204 Endsem Problems

Program 1. Write a program to read in a directed graph from the user and print its reverse graph, i.e. the graph with all the edges reversed. Your program should further check if the reversed graph has a cycle. In case a cycle exists, print the sequence of vertices that form a cycle. (If the graph has more than one cycle, print any one of them.) You may assume that the input graph is connected, i.e. for any two vertices **u** and **v** there is either a path from **u** to **v** or from **v** to **u**.

Use the **adjacency list** representation to store the graph. You can **NOT** use Python's inbuilt list data structure to store the list of adjacent vertices of a given vertex. So you will have to create a singly linked list of adjacent vertices for every vertex of the graph.



A sample run for the above directed graph is given below.

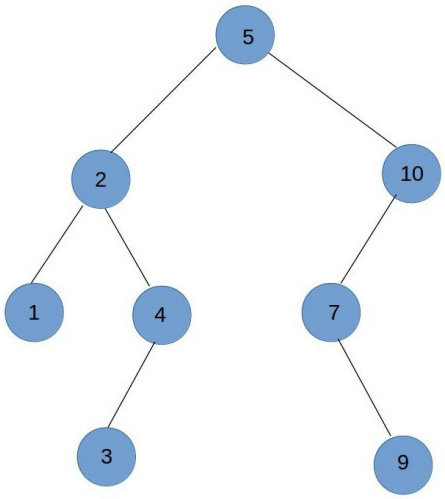
Sample Run:

Enter the number of vertices and edges : 6 10
Enter the edges: 0 1
0 2
0 3
1 2
2 4
3 1
4 1
4 5
5 1
5 3

The reversed graph's adjacency list representation is:
Vertex 0:
Vertex 1: 0 3 4 5
Vertex 2: 0 1
Vertex 3: 0 5
Vertex 4: 2
Vertex 5: 4

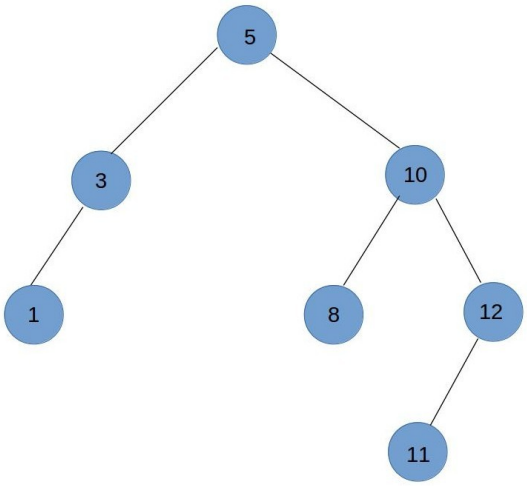
Cycle detected. Vertices: 1,2,4,5,1

Program 2. Write a program to do insertions in a BST and print the Reverse Level-Order traversal of the resulting tree. For e.g., for the BST shown below, the reverse level-order traversal will print: 3 9 1 4 7 2 10 5.



Build your tree using a sequence of BST insert operations. For e.g. the BST above can be built by the sequence of inserts: 5 2 1 4 3 10 7 9. Read in the sequence of keys to be inserted from the user.

A sample run of the program (for the different BST shown below) is given.



Sample Run:
Enter the keys to be inserted in order: 5 3 1 10 8 12 11.

The reverse order traversal of the BST is: 11 1 8 12 3 10 5.