1 Question 1

The DeepWalk architecture can be extended to directed graphs by leveraging the `.successors()` function in the Networkx library to account for successor nodes. For weighted graphs, it can be adapted by defining a probability distribution over the edges, where the selection probabilities are proportional to the edge weights.

2 Question 2

The embeddings X_1 and X_2 are related by a reflection about the x-axis. This transformation preserves the graph's symmetry, ensuring the following:

- The triangular structures formed by nodes (v_1, v_2, v_3) and (v_4, v_5, v_6) remain unchanged.
- The roles of v_3 and v_4 as central connecting nodes in their respective clusters are maintained.

This reflection arises naturally from the random nature of the DeepWalk algorithm. While the embeddings are generated by the same underlying process, the randomness (e.g., random walks or initialization) can result in transformations such as reflections, rotations, or translations. These transformations are symmetry-preserving and do not affect the structural or topological properties of the graph encoded in the embeddings.

3 Question 3

The size of the *receptive field* of a node in a Graph Convolutional Network (GCN) refers to the maximal distance (in terms of the number of edges) from the given node to other nodes whose features contribute to the computation of the prediction for that node.
In the specific case of Task 10, the GCN has exactly 2 message-passing layers. Each message-passing layer allows a node to aggregate information from its immediate neighbors (i.e., nodes at a distance of 1 edge). After the first layer, a node will have access to the features of its direct neighbors. After the second layer, a node can aggregate information from nodes up to 2 edges away, because it will receive information from the neighbors of its neighbors.
Conclusion: In this architecture, the maximum receptive field of any node is limited to a distance of 2 edges. This means that the prediction for a given node takes into account the features of all nodes within 2 hops from it in the graph.
Mathematical Summary: For a GCN with k message-passing layers:

$$\text{Receptive Field} = k$$

Since Task 10 specifies 2 message-passing layers:

$$\text{Receptive Field} = 2$$

In this setup, for any node v, the prediction Y_v will be influenced by the features of: - The node v itself (distance 0), - Its immediate neighbors (distance 1), - The neighbors of its neighbors (distance 2).

4 Question 4

Table 1 below shows all the matrices required for the computation of Z^1 for the two graphs: K_4 and S_4. Below, we provide a few key points regarding the computation of these matrices:

- A node is never connected to itself in the adjacency matrix A.
- The central node in S_4 is chosen to be the first node in the adjacency matrix.
- The feature matrix X is initialized as a vector of ones.

The rows of the matrix Z^1 represent the embeddings of the corresponding nodes in the graph. Observing the structure of Z^1, we notice that nodes with identical structural roles (e.g., all peripheral nodes in S_4 or all nodes in K_4) have the same activation values. This is due to the symmetry in both the feature matrix X and the adjacency structure.
Moreover, for K_4, all nodes are structurally identical, leading to uniform embeddings. For S_4, the central node has a distinct embedding, while the peripheral nodes have identical embeddings.
Additionally, we notice that the first coordinate of the embeddings is consistently zero. This is a direct result of applying the ReLU activation function, which sets all negative values to zero.
If the node features X were sampled randomly (e.g., from a random uniform distribution), the resulting embeddings in Z^1 would reflect the variability in the input features. Consequently, the repeating patterns observed in Z^1 for K_4 and S_4 would no longer be present.

K_4	S_4
$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$	$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
$\hat{A} = \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$	$\hat{A} = \begin{pmatrix} 0.25 & 0.35 & 0.35 & 0.35 \\ 0.35 & 0.5 & 0 & 0 \\ 0.35 & 0 & 0.5 & 0 \\ 0.35 & 0 & 0 & 0.5 \end{pmatrix}$
$Z^0 = \begin{pmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{pmatrix}$	$Z^0 = \begin{pmatrix} 0 & 0.65 \\ 0 & 0.43 \\ 0 & 0.43 \\ 0 & 0.43 \end{pmatrix}$
$Z^1 = \begin{pmatrix} 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \end{pmatrix}$	$Z^1 = \begin{pmatrix} 0 & 0.37 & 0.31 \\ 0 & 0.27 & 0.22 \\ 0 & 0.27 & 0.22 \\ 0 & 0.27 & 0.22 \end{pmatrix}$

Table 1: Computation of Z^1 for K_4 and S_4.

5 4.3 Node Classification and Representation on Cora

In this task, we apply a Graph Neural Network (GNN) to the Cora dataset, which includes both the graph structure and node features. The results obtained for the node classification task are summarized in Table 2:

Metric	GNN
Test Loss	0.5895
Test Accuracy	86.35%

Table 2: Performance metrics for node classification on the Cora dataset using a GNN.

Figure 1 shows the visualization of the node embeddings obtained using the GNN, projected onto a 2D space using t-SNE. Each node is classified into its respective class, and the visualization highlights the quality of the classification.
The results indicate that the GNN achieves strong classification performance with a test accuracy of 86.35%. The embeddings demonstrate clear separations between different classes in the dataset, confirming the effectiveness of the GNN in leveraging both node features and graph structure for the task.

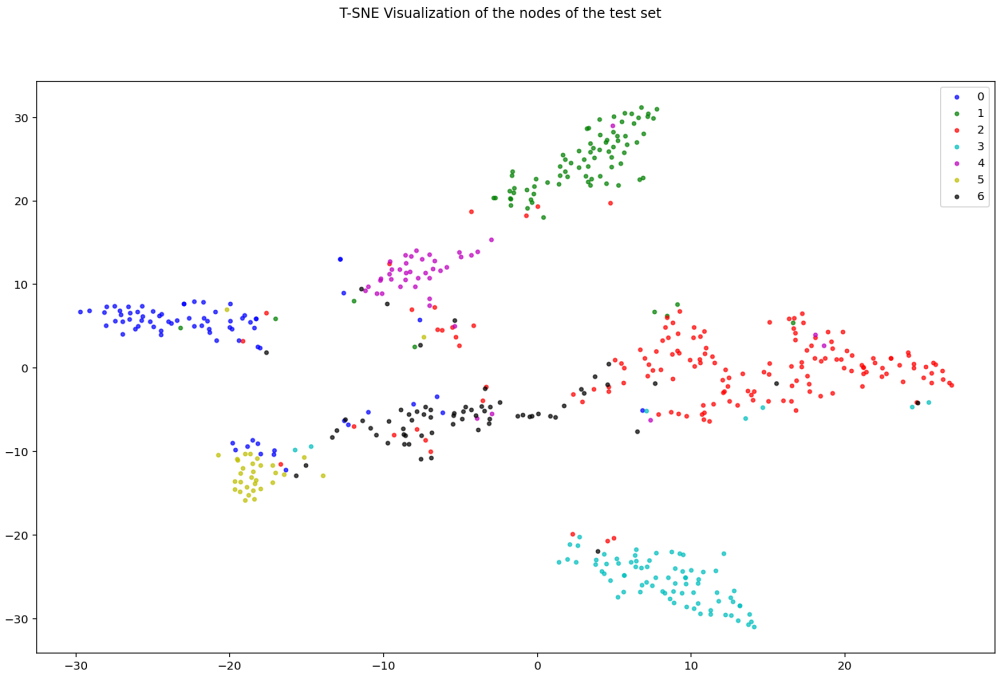

Figure 1: Node classification results on the Cora dataset using a GNN and t-SNE for visualization.