Program:

```
# Depth First Search (DFS) implementation for a warehouse graph

# Sample warehouse graph as an adjacency list

warehouse_graph = {

  'A': ['B', 'C'],

  'B': ['D', 'E'],

  'C': ['F'],

  'D': [],

  'E': ['F'],

  'F': []
```

```
}

# Function to perform DFS
def dfs(graph, start, goal, visited=None, path=None):
    if visited is None:
        visited = set()
    if path is None:
        path = []
    # Mark current node as visited and add to path
    visited.add(start)
    path.append(start)
    # If goal is found, return the path
    if start == goal:
        return path

    # Explore neighbors
    for neighbor in graph[start]:
        if neighbor not in visited:
            result = dfs(graph, neighbor, goal, visited, path[:])  # Use path[:] to copy path
            if result:  # Stop if a path is found
                return result
        return None  # No path found
# Example usage
start_node = 'A'
goal_node = 'F'
```

```
path_found = dfs(warehouse_graph, start_node, goal_node)
print(f"DFS Path from {start_node} to {goal_node}: {path_found}")
```

Output:

```
DFS Path from A to F: ['A', 'B', 'E', 'F']
```

Or

```
DFS Path from A to F: ['A', 'C', 'F']
```

Result:

Thus the given case-based discussion program has been Implemented successfully and the program has been uploaded in Github link.