```
def solve_pyramid_descent(pyramid, target):
    def find_path(row, col, current_product, path):
        if row == len(pyramid):
            if current_product == target:
                return path
            return None

        # Explore the left and right paths
        left_path = find_path(row + 1, col, current_product *
pyramid[row][col], path + "L")
        if left_path:
            return left_path

        right_path = find_path(row + 1, col + 1, current_product *
pyramid[row][col], path + "R")
        if right_path:
            return right_path

        return None

    return find_path(0, 0, 1, "")

# Parse the pyramid from the file
def parse_pyramid(file_path):
    try:
        with open(file_path, "r") as file:
            lines = file.readlines()

        target = int(lines[0].split(":")[1].strip())
        pyramid = []

        for line in lines[1:]:
            pyramid.append([int(x) for x in line.split(",")])

        return pyramid, target
    except FileNotFoundError:
        print(f"Error: File not found at {file_path}")
        return None, None

# Main execution
```

```python
if __name__ == "__main__":
    file_path = "pyramid_sample_input.txt"  # Update this path
    pyramid, target = parse_pyramid(file_path)

    if pyramid and target:
        result = solve_pyramid_descent(pyramid, target)
        if result:
            print(f"Path to achieve the target ({target}): {result}")
        else:
            print(f"No path found to achieve the target ({target}).")
    else:
        print("Ensure the input file is correctly placed and accessible.")
```