Shanon Fano Encoding

```
# Shannon-Fano Encoding (Binary)



# -------- FIND BEST SPLIT --------
def FindBestSplit(prob):

    min_diff = float("inf")
    best_k = 1

    for k in range(1, len(prob)):

        left_sum = sum(prob[:k])
        right_sum = sum(prob[k:])

        diff = abs(left_sum - right_sum)

        if diff < min_diff:
            min_diff = diff
            best_k = k

    return best_k



# -------- ENCODING FUNCTION --------
def encoding(prob, idx, code):

    if len(idx) == 1:
```

```
    return code


if len(idx) == 2:

    code[idx[0]] += "0"
    code[idx[1]] += "1"

    return code


best_k = FindBestSplit(prob)


left_idx = idx[:best_k]
right_idx = idx[best_k:]


for i in left_idx:
    code[i] += "0"

for i in right_idx:
    code[i] += "1"


code = encoding(prob[:best_k], left_idx, code)
code = encoding(prob[best_k:], right_idx, code)
```

```
    return code



# -------- MAIN --------

# New example probabilities
prob = [0.35, 0.25, 0.15, 0.10, 0.08, 0.07]

N = len(prob)



# Initialize codes (1-based indexing)
code = [""] * (N + 1)



# Sort in descending order
temp = list(zip(prob, range(1, N + 1)))

temp.sort(reverse=True, key=lambda x: x[0])

SortedProb, order = zip(*temp)

SortedProb = list(SortedProb)
order = list(order)



# Run encoding
```

```
code = encoding(SortedProb, order, code)




# -------- OUTPUT TABLE --------


print("\nShannon-Fano Encoding Table\n")


print("+--------+-------------+-----------+")
print("| Symbol | Probability |   Code    |")
print("+--------+-------------+-----------+")


for i in range(1, N + 1):


    print(f"| {i:^6} | {prob[i-1]:^11.2f} | {code[i]:^9} |")


print("+--------+-------------+-----------+")
```

Shannon-Fano Encoding Table

Symbol	Probability	Code
1	0.35	00
2	0.25	01
3	0.15	100
4	0.10	101
5	0.08	110
6	0.07	111