1 Problem Statement

2 Code

```
# Project Euler #622 | Sidharth Baskaran | 01/15/2022

import time

# function that performs the out-shuffle for visualization (not used in solution)
def permute_list(deck):
    n = len(deck)
    copy = [0] * n
    for i in range(1, n - 1):
        copy[2*i % (n - 1)] = deck[i]
    copy[0] = deck[0]
    copy[n-1] = deck[n-1]
    return copy

# permutation visualization (not used in solution)
def sf(n):
    deck = [*range(n)]
    copy = deck
    perms = 0
    while True:
        copy = permute_list(deck)
        perms += 1
        if copy == deck:
            return perms

def solve(k):
    upper = 2 ** k - 1
    cache = [2 ** x - 1 for x in range(k - 1, 0, -1)]
    total = 0
    # finding divisors of 2^k - 1
    for i in range(upper, 1, -1):
        if upper % i == 0:
            # excluding divisors that work for values smaller than k
            # (i.e. values of n where deck can be recreated in less than 60 shuffles)
            flag = False
            for p in cache:
                if p % i == 0:
                    flag = True
                    break
            if not flag:
                total += i + 1
    print(total)

if __name__ == "__main__":
    s = time.time()
    solve(60)
    e = time.time()

    print('%.3fms' % ((e-s)*1000))
```

3 Explanation