run24()

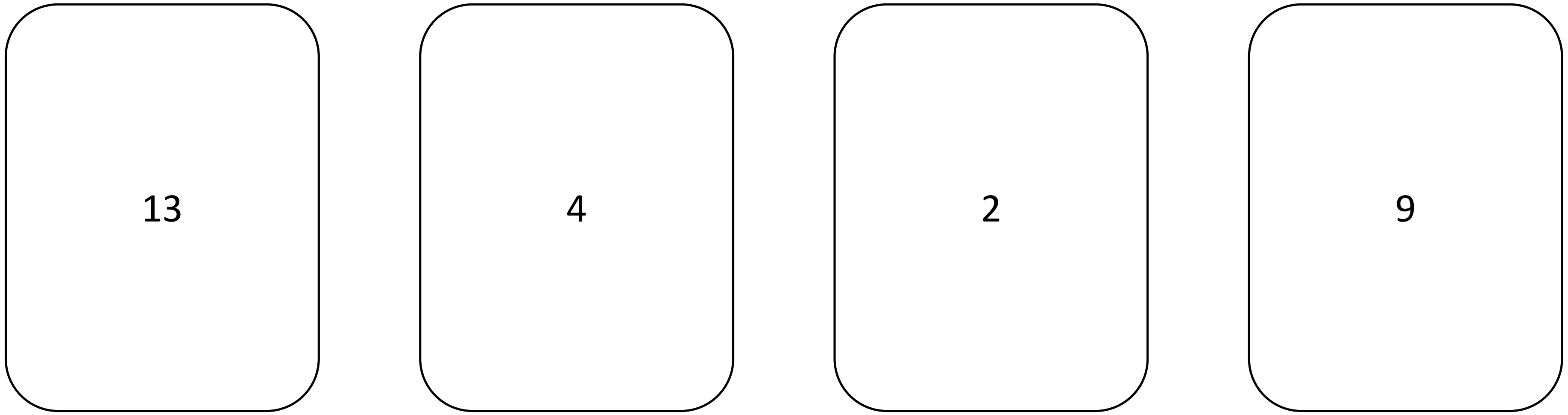
def introduction():

```
print(“24 is a mathematics game in poker.”)
print(“An interesting kill-time & social activity”)
print(“All you need is a deck of cards (Now a laptop)”)
```

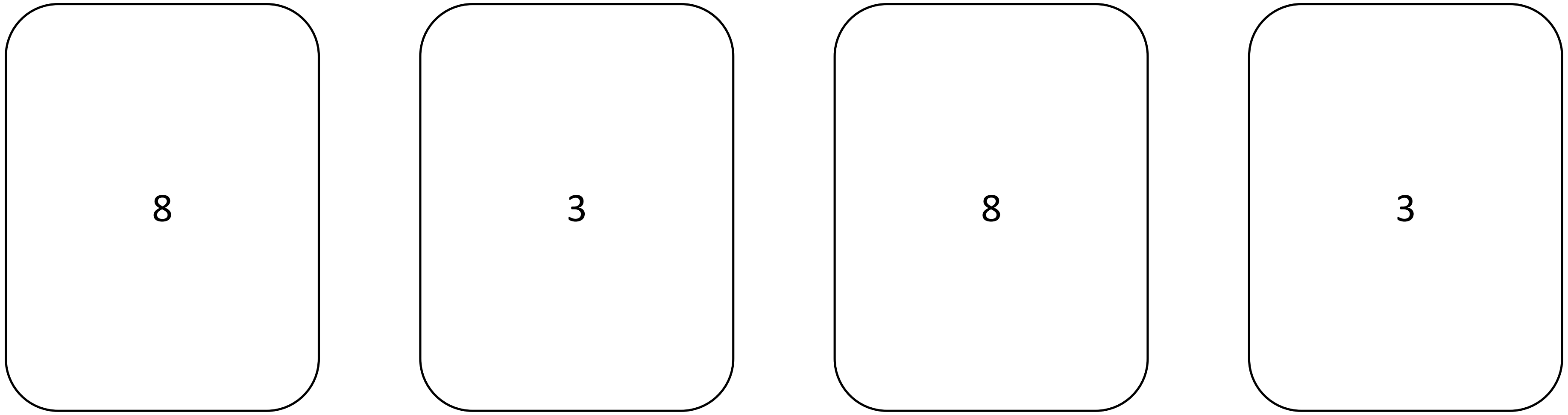
def rules(players):

```
cards=generate4Cards() #from a deck
brainstorm(brains) #use + - * / to get 24
for player in players:
    if player.shout == True and player.solution(cards)==24:
        player.win=True
```

haveSomeFun()



If you encounter a difficult one…

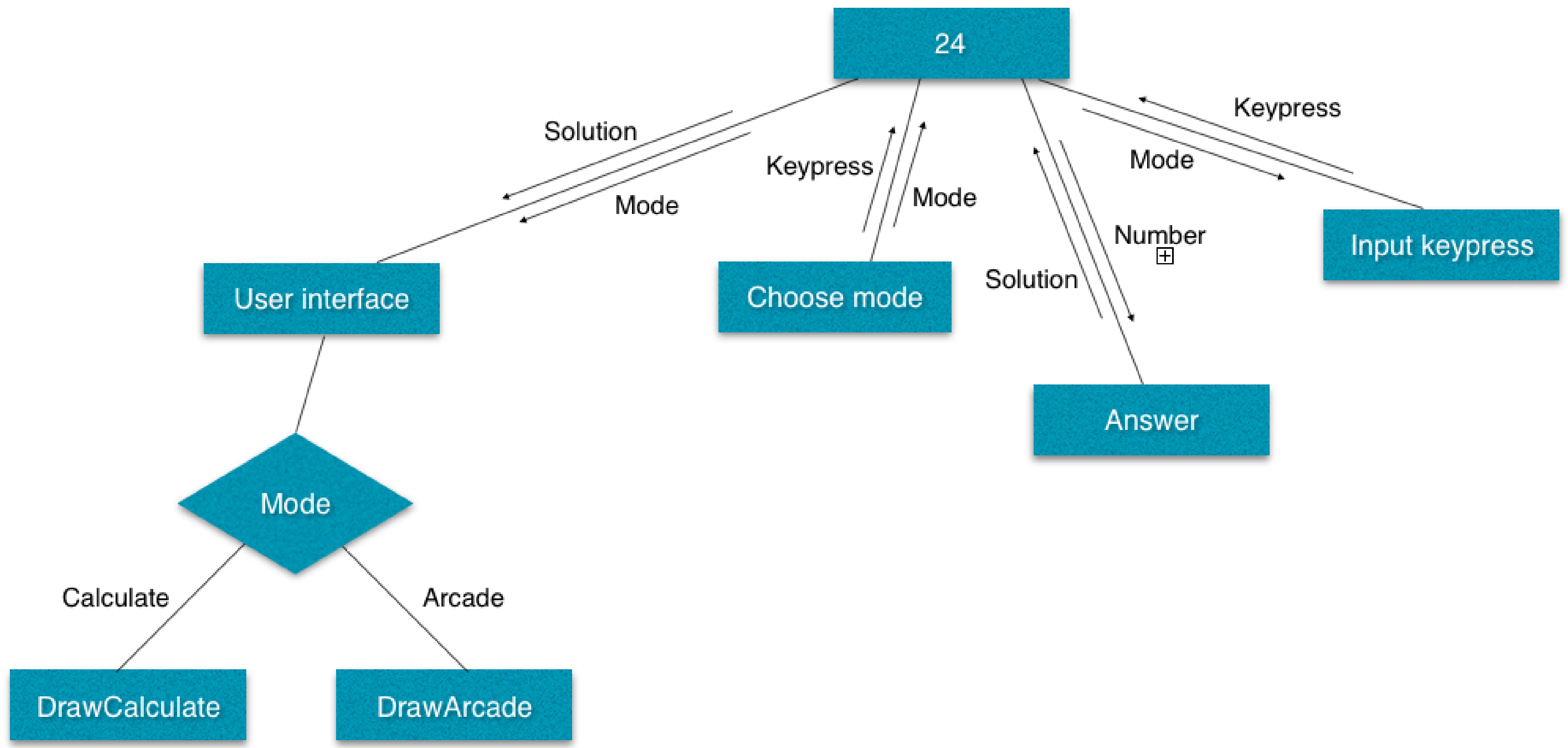


1. Try to solve it yourself
2. Look it up online
3. … (You are a 闲得蛋疼的CS student)

Make a python program!

- An algorithm to solve 24
- A concise but not simple UI
- An interesting arcade mode for fun besides only solving 24

Structure


24
Solution
Mode
Keypress
Mode
Keypress
Mode
Number
Solution
User interface
Choose mode
Input keypress
Answer
Mode
Calculate
Arcade
DrawCalculate
DrawArcade

Explanation of the algorithm

4 7 8 3

3	+	4	-	7	*	8
7	-	3	*	4	+	8

Explanation of the algorithm

3 4 7 8 | 3 4 8 7 | 3 7 4 8 | … (4! sets)

+ + + + | + + + - | + / * - | … (4^4 set)

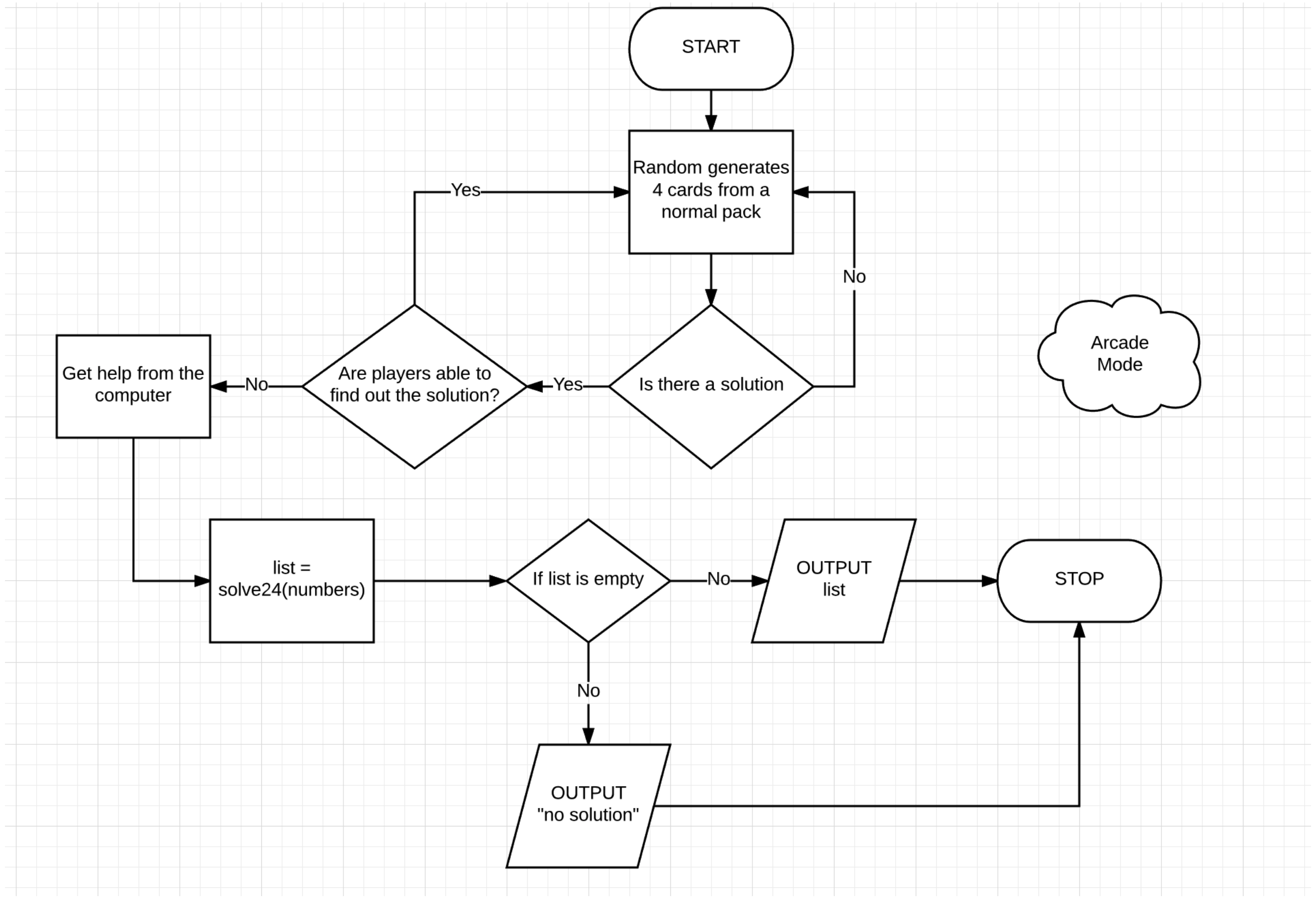
Expressions

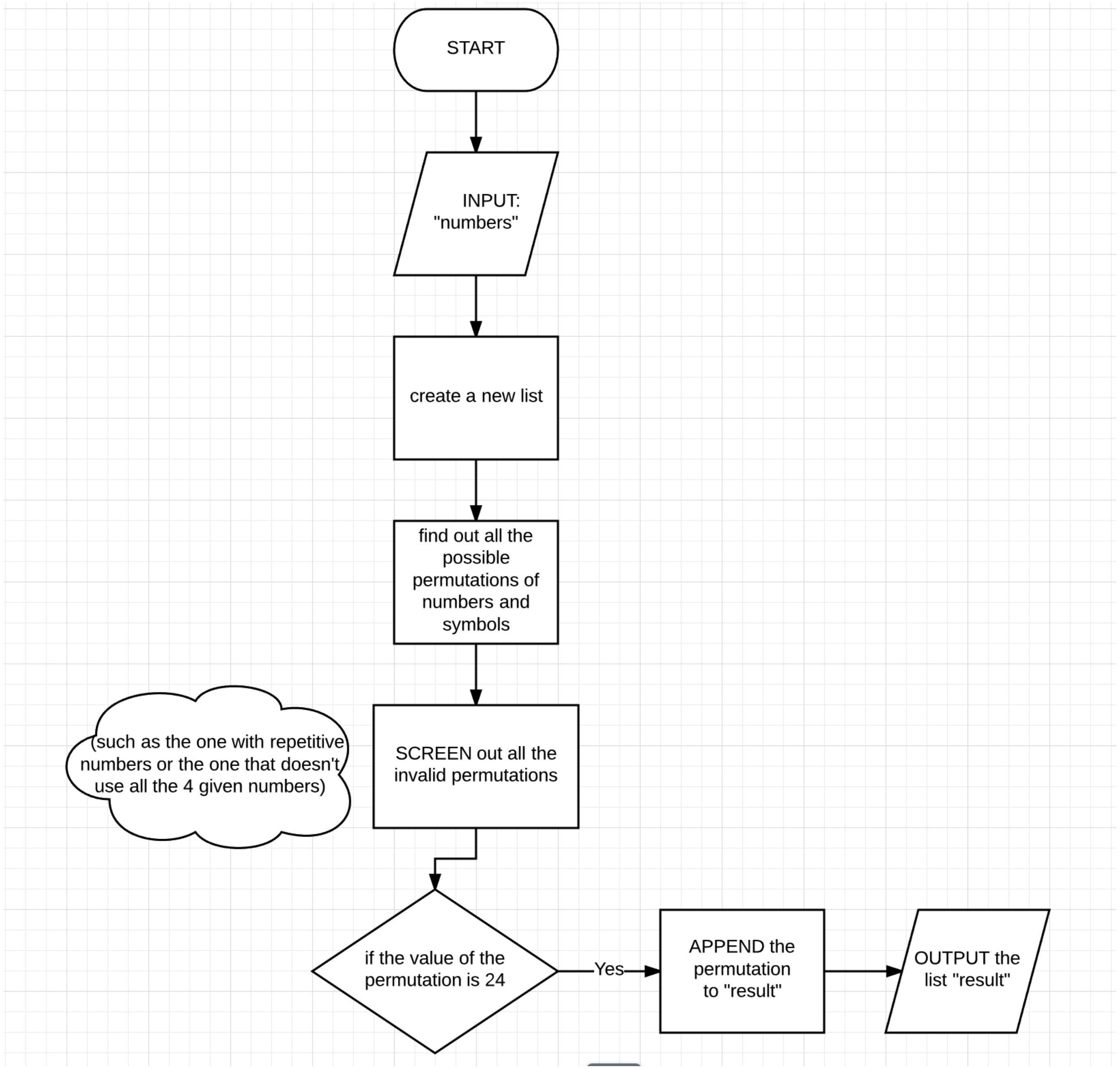
1. ((num1 sign1 num2) sign2 num3) sign3 num4
2. (num1 sign1 num2) sign2 (num3 sign3 num4)
3. (num1 sign1 (num2 sign2 num3)) sign3 num4
4. num1 sign1 ((num2 sign2 num3) sign3 num4)
5. num1 sign1 (num2 sign2 (num3 sign3 num4))

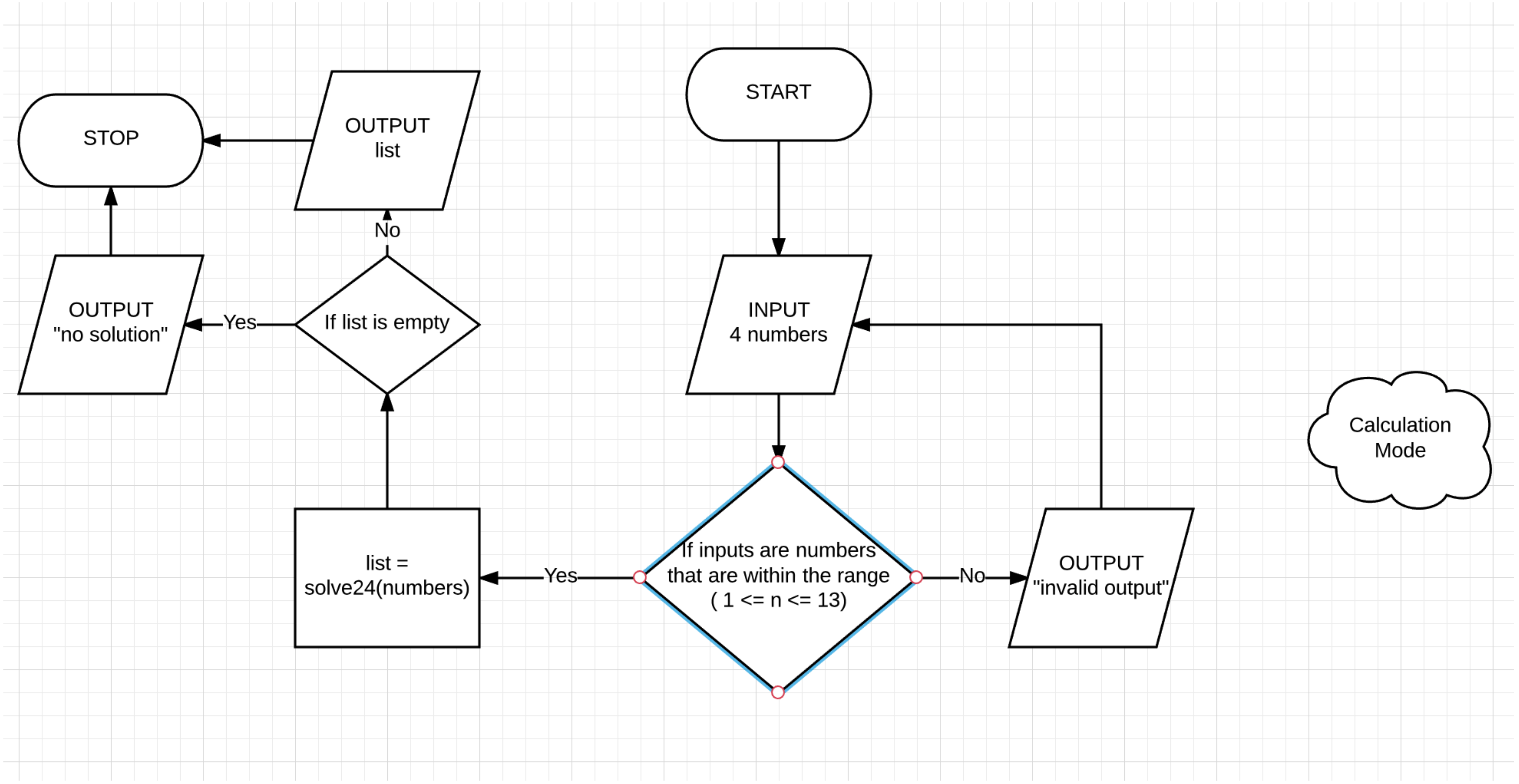
$$4! \times 4^4 \times 5 = ?$$

30720 possibilities

Flowcharts (additional)




START
INPUT: "numbers"
create a new list
find out all the possible permutations of numbers and symbols
SCREEN out all the invalid permutations
(such as the one with repetitive numbers or the one that doesn't use all the 4 given numbers)
if the value of the permutation is 24
Yes
APPEND the permutation to "result"
OUTPUT the list "result"


START
INPUT
4 numbers
If inputs are numbers
that are within the range
(1 <= n <= 13)
No
OUTPUT
"invalid output"
Yes
list =
solve24(numbers)
If list is empty
Yes
OUTPUT
"no solution"
No
OUTPUT
list
STOP
Calculation
Mode