

Laporan Tugas Kecil 1

Strategi Algoritma (IF-2211)

Cryptarithmic

Frederic Ronaldi / 13519134

Algoritma Brute Force :

1. Menggabungkan string-string operand menjadi satu string utuh tanpa spasi, lalu gabungan string tersebut diubah ke list. List tersebut diubah ke set untuk menghilangkan karakter-karakter yang duplikat. Set tersebut diubah kembali ke list untuk diproses di tahap selanjutnya.
2. Membuat 2 list baru yang identik berisi angka 0 hingga panjang karakter unik - 1. List pertama selanjutnya akan disebut sebagai list angka. List kedua selanjutnya akan disebut sebagai list permutasi. Sebagai contoh:
List karakter unik : ['A', 'B', 'C', 'D', 'E']
List angka : [0, 1, 2, 3, 4]
List permutasi : [0, 1, 2, 3, 4]
3. List permutasi diproses menjadi dictionary untuk menandakan angka-angka pada setiap karakter unik. Sebagai contoh, pada contoh diatas akan dibuat dictionary sebagai berikut: {'A' : 0, 'B' : 1, 'C' : 2, 'D' : 3, 'E' : 4}
4. Dictionary tersebut akan diproses dengan mengganti masing-masing karakter pada operand dan hasil dengan angka-angka yang ada di dictionary tersebut. Kemudian, dilakukan pengecekan apabila jawaban tersebut benar, yaitu penjumlahan setiap angka integer pada operand harus sama dengan angka dalam hasil.
5. Apabila dictionary dari list permutasi tersebut bukan merupakan solusi, maka akan dicari list permutasi lain yang memungkinkan. List permutasi tersebut akan diganti dengan list permutasi selanjutnya yang diurutkan secara lexicographical. Sebagai contoh, permutasi selanjutnya list permutasi [0, 1, 2, 3, 4] adalah [0, 1, 2, 4, 3]. Lalu selanjutnya secara berurut adalah [0, 1, 3, 2, 4], [0, 1, 3, 4, 2], [0, 1, 4, 2, 3], dan seterusnya hingga permutasi terakhir yaitu [4, 3, 2, 1, 0].
6. Setiap mendapatkan list permutasi selanjutnya, step ketiga dan keempat akan diulangi kembali hingga permutasi dengan list permutasi tersebut habis.
7. Jika list permutasi dari list angka tersebut sudah habis, maka akan dicari kombinasi list angka selanjutnya yang memungkinkan. List angka tersebut akan diganti dengan list angka selanjutnya yang memiliki angka yang berbeda dengan list angka sebelumnya secara terurut menaik. Sehingga jika dipraktikan dapat diperoleh contoh sebagai berikut, list angka selanjutnya dari list angka [0, 1, 2, 3, 4] adalah [0, 1, 2, 3, 5]. Lalu selanjutnya secara berurut adalah [0, 1, 2, 3, 6], [0, 1, 2, 3, 7], [0, 1, 2, 3, 8], [0, 1, 2, 3, 9], [0, 1, 2, 4, 5], [0, 1, 2, 4, 6], dan seterusnya hingga list angka terakhir yaitu [5, 6, 7, 8, 9].

8. Setiap mendapatkan list angka selanjutnya, list permutasi akan kembali diisi dengan list yang identik dengan list angka selanjutnya, lalu step ketiga hingga ketujuh akan diulangi kembali hingga kombinasi list angka habis.
9. Jika saat list angka dan setiap list permutasi dari list angka tersebut telah habis dicek, solusi masih belum ditemukan. Maka dapat dipastikan bahwa persamaan tersebut tidak memiliki solusi.

Source Code dalam bahasa Python

```
1  import time
2
3
4  def maxCharacter(operands, result):
5      result = len(result[0])
6      for x in operands:
7          if (result < len(x)):
8              result = len(x)
9      return result
10
11
12 def printEquation(operands, result):
13     maxChar = maxCharacter(operands, result)
14     for i in range(len(operands)):
15         spaceCount = maxChar - len(operands[i])
16         if (i == len(operands) - 1):
17             print(' ' * spaceCount + operands[i] + '+')
18         else:
19             print(' ' * spaceCount + operands[i])
20     print("-" * maxChar)
21     spaceCount = maxChar - len(result[0])
22     print(' ' * spaceCount + result[0])
23     print('')
24
25
26 def isValidData(operands, result, data):
27     startChar = []
28     for x in operands:
29         startChar.append(x[0])
30     startChar.append(result[0][0])
31     # convert to set to remove duplicate
32     startChar = set(startChar)
33     for c in startChar:
34         if (int(data[c]) == 0):
35             return False
36     return True
37
```

```

39 def check(operands, result, data):
40     intOperands = []
41     intResult = []
42
43     if (not(isValidData(operands, result, data))):
44         return False
45
46     for x in operands:
47         operand = ''
48         for c in x:
49             operand += str(data[c])
50         intOperands.append(int(operand))
51
52     res = ''
53     for c in result[0]:
54         res += str(data[c])
55     intResult.append(int(res))
56
57     res = 0
58     for x in intOperands:
59         res += x
60
61     return res == intResult[0]
62
63
64 def generateOperandsAndResult(operands, result, data):
65     intOperands = []
66     intResult = []
67
68     for x in operands:
69         operand = ''
70         for c in x:
71             operand += str(data[c])
72         intOperands.append(operand)
73
74     res = ''
75     for c in result[0]:
76         res += str(data[c])
77     intResult.append(res)
78
79     return [intOperands, intResult]
80

```

```

82  def nextNumbers(arr, i):
83      if (int(arr[i]) + 1 <= 9 - (len(arr) - 1) + i):
84          arr[i] = str(int(arr[i]) + 1)
85          return True
86      else:
87          while (int(arr[i]) + 1 > 9 - (len(arr) - 1) + i and i > 0):
88              i -= 1
89
90          if (i != 0 or int(arr[i]) + 1 <= 9 - (len(arr) - 1) + i):
91              arr[i] = str(int(arr[i]) + 1)
92              while (i < len(arr) - 1 and int(arr[i]) + 1 <= 9 - (len(arr) - 1) + i + 1):
93                  arr[i + 1] = str(int(arr[i]) + 1)
94                  i += 1
95              return True
96          else:
97              return False
98
99
100 def nextPermutation(arr):
101     # find pivot
102     pivotIndex = -1
103
104     for i in range(len(arr), 1, -1):
105         if (int(arr[i - 1]) > int(arr[i - 2])):
106             pivotIndex = i - 2
107             break
108
109     if (pivotIndex == -1):
110         # arr is the last permutation
111         return False
112
113     swapIndex = -1
114     # find the rightmost successor to pivot in suffix
115     for i in range(len(arr), pivotIndex + 1, -1):
116         if (int(arr[i - 1]) > int(arr[pivotIndex])):
117             swapIndex = i - 1
118             break
119
120     arr[pivotIndex], arr[swapIndex] = arr[swapIndex], arr[pivotIndex]
121
122     suffix = []
123     for i in range(len(arr), pivotIndex + 1, -1):
124         suffix.append(arr[i - 1])
125
126     for i in range(len(arr)):
127         if (i > pivotIndex):
128             arr[i] = suffix[i - pivotIndex - 1]
129
130     return True
131

```

```

133 def generateData(arr, characters):
134     data = {}
135     for c in characters:
136         data[c] = arr[0]
137         arr = arr[1:]
138     return data
139
140
141 def solver(operands, result):
142     joinedString = ''
143     for i in operands:
144         joinedString += i
145     joinedString += result[0]
146     # get unique alphabet in operands and result
147     characters = list(set(joinedString))
148     if (len(characters) > 10):
149         print("Tidak punya solusi")
150         return
151
152     startNumber = [i for i in range(len(characters))]
153     cursor = len(startNumber) - 1
154     startPermutation = list(startNumber)
155
156     count = 1
157     data = generateData(startPermutation, characters)
158     if (check(operands, result, data)):
159         correct = generateOperandsAndResult(operands, result, data)
160         printEquation(correct[0], correct[1])
161         return count
162     else:
163         while (nextPermutation(startPermutation)):
164             count += 1
165             data = generateData(startPermutation, characters)
166             if (check(operands, result, data)):
167                 correct = generateOperandsAndResult(operands, result, data)
168                 printEquation(correct[0], correct[1])
169                 return count
170
171         while (nextNumbers(startNumber, cursor)):
172             count += 1
173             startPermutation = list(startNumber)
174             data = generateData(startPermutation, characters)
175             if (check(operands, result, data)):
176                 correct = generateOperandsAndResult(operands, result, data)
177                 printEquation(correct[0], correct[1])
178                 return count
179             else:
180                 while (nextPermutation(startPermutation)):
181                     count += 1
182                     data = generateData(startPermutation, characters)
183                     if (check(operands, result, data)):
184                         correct = generateOperandsAndResult(
185                             operands, result, data)
186                         printEquation(correct[0], correct[1])
187                         return count
188
189     return count

```

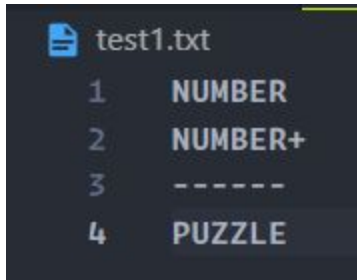
```

191
192 # main program
193 fname = str(input("Masukkan nama file: "))
194 f = open(fname)
195 print("")
196 f1 = f.readlines()
197
198 operands = []
199 result = []
200
201 failure = False
202
203 operandFinished = False
204 for i in f1:
205     isDelimiter = '-' in i
206     isLastOperand = '+' in i
207
208     # remove space
209     i = i.replace(" ", "")
210     # remove enter
211     i = i.rstrip("\n")
212
213     isEmptyString = len(i) == 0
214
215     if (isLastOperand):
216         i = i.replace("+", "")
217
218     if (not(isDelimiter) and not(isEmptyString)):
219         # add to array
220         if (operandFinished):
221             result.append(i)
222
223         # check for failure
224         if (len(f1) < 4 or len(result) != 1 or len(operands) == 0):
225             failure = True
226
227         if (failure):
228             print ("Input format wrong")
229             break
230         else:
231             # timer start
232             startTime = time.time()
233             printEquation(operands, result)
234             count = solver(operands, result)
235             # timer end
236             endTime = time.time()
237             print("Total waktu: {:.3f} detik".format(endTime-startTime))
238             print("Total tes: {:d}".format(count))
239             print("")
240
241             operandFinished = False
242             operands = []
243             result = []
244
245         else:
246             operands.append(i)
247
248     # check if last operand
249     if (isLastOperand):
250         operandFinished = True
251

```

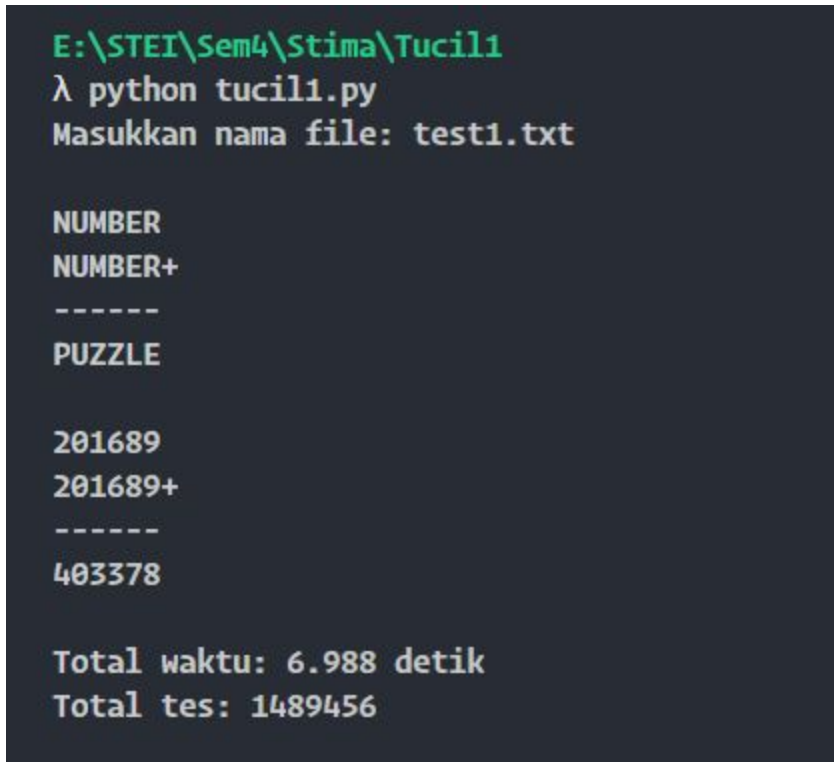
Screenshot Program

1. Input



```
test1.txt
1    NUMBER
2    NUMBER+
3    -----
4    PUZZLE
```

Output



```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test1.txt

NUMBER
NUMBER+
-----
PUZZLE

201689
201689+
-----
403378

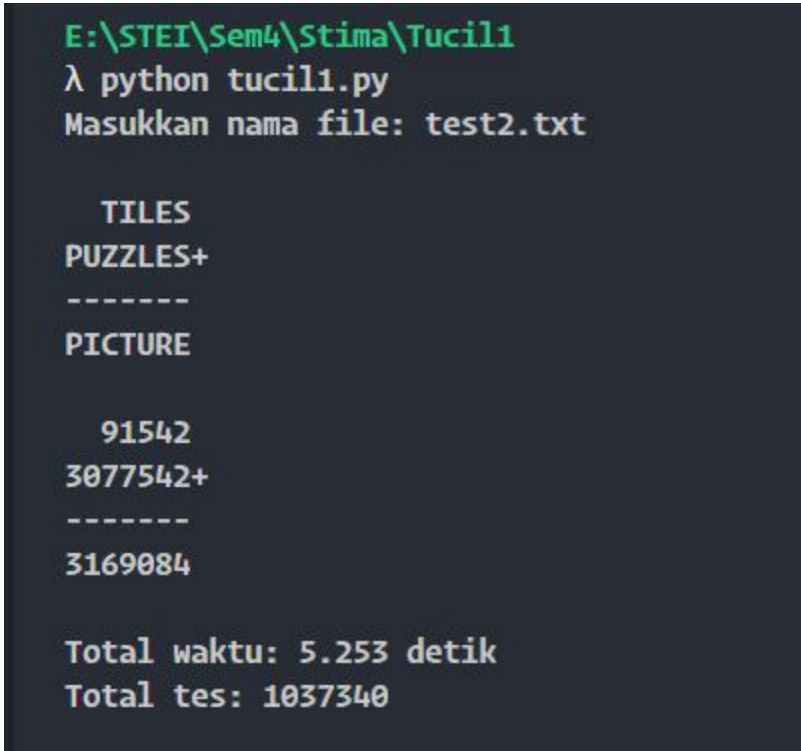
Total waktu: 6.988 detik
Total tes: 1489456
```


2. Input



```
test2.txt
1 |  TILES
2 |  PUZZLES+
3 |  -----
4 |  PICTURE
```

Output



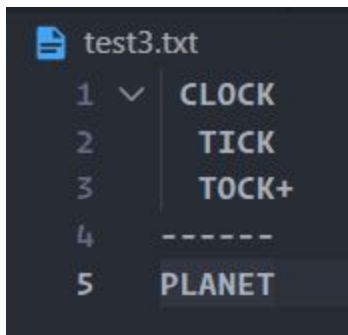
```
E:\STET\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test2.txt

    TILES
    PUZZLES+
    -----
    PICTURE

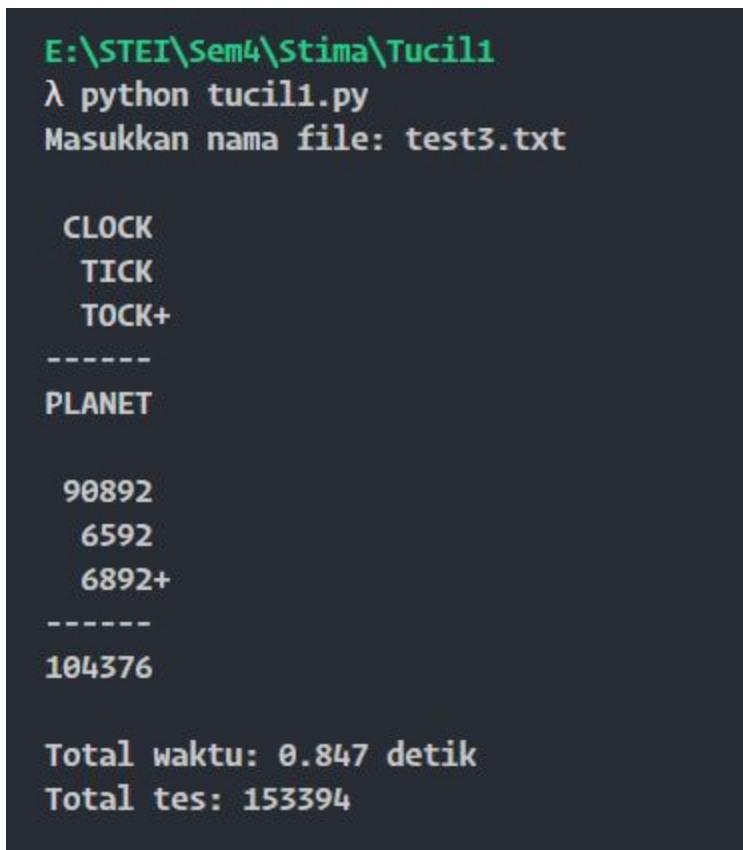
    91542
    3077542+
    -----
    3169084

Total waktu: 5.253 detik
Total tes: 1037340
```

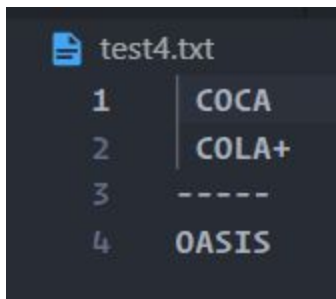

3. Input



Output



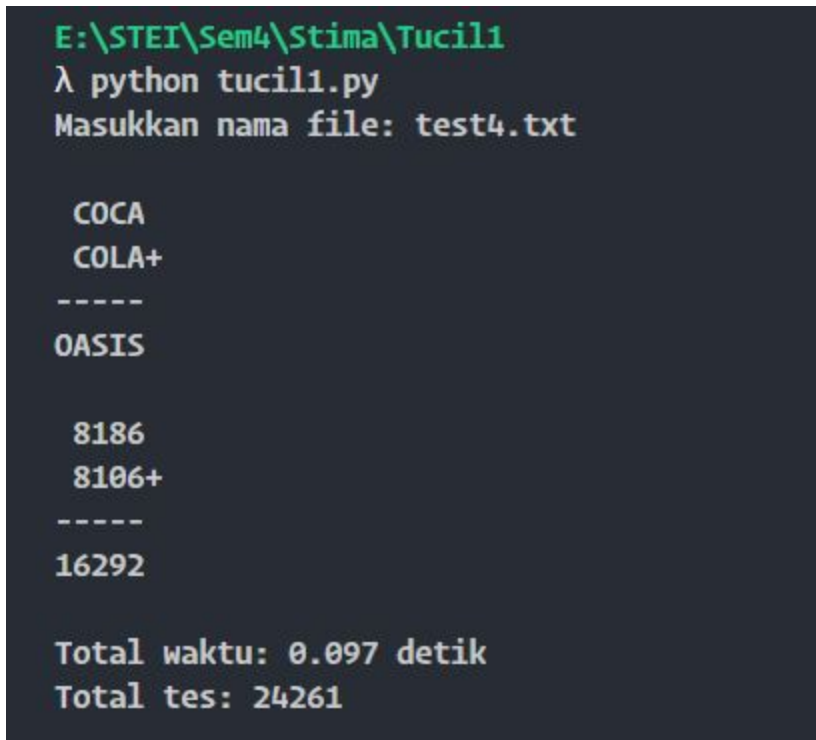
4. Input



A screenshot of a text editor showing a file named 'test4.txt'. The file contains four lines of text, each preceded by a number from 1 to 4. The text is as follows:

No	Brand
1	COCA
2	COLA+
3	-----
4	OASIS

Output



A screenshot of a terminal window showing the execution of a Python script. The prompt is 'E:\STEI\Sem4\Stima\Tucil1'. The user enters 'python tucil1.py'. The prompt then changes to 'Masukkan nama file: test4.txt'. The script then outputs the contents of the file, followed by a calculation of the total time and the total number of tests.

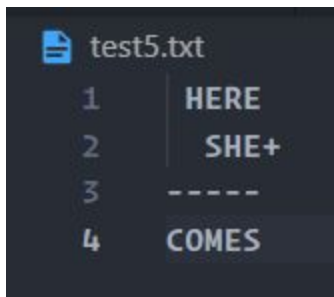
```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test4.txt

COCA
COLA+
-----
OASIS

8186
8106+
-----
16292

Total waktu: 0.097 detik
Total tes: 24261
```

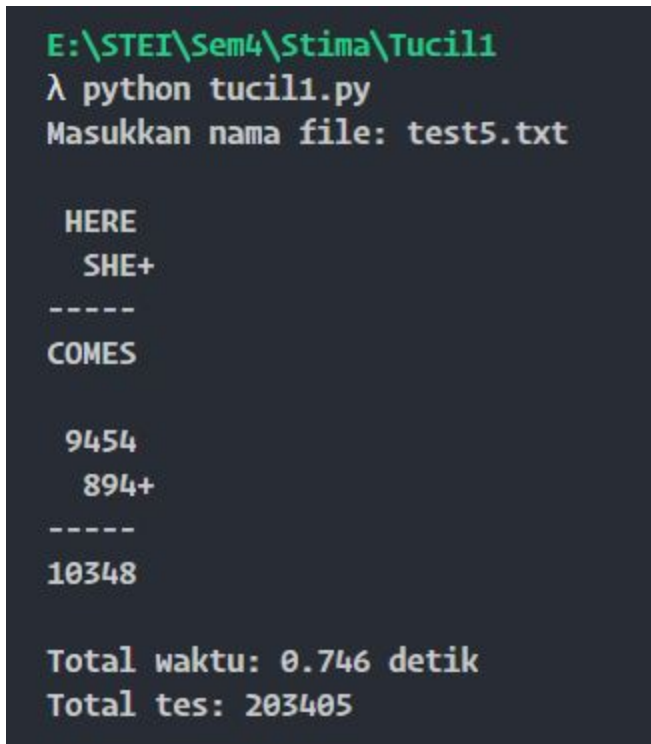
5. Input



test5.txt

1	HERE
2	SHE+
3	-----
4	COMES

Output



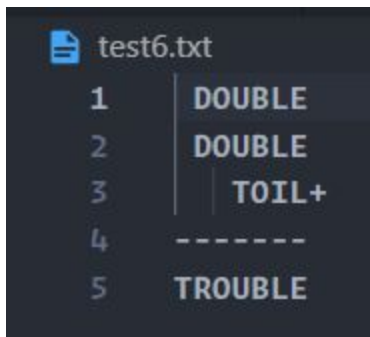
```
E:\STET\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test5.txt

HERE
SHE+
-----
COMES

9454
894+
-----
10348

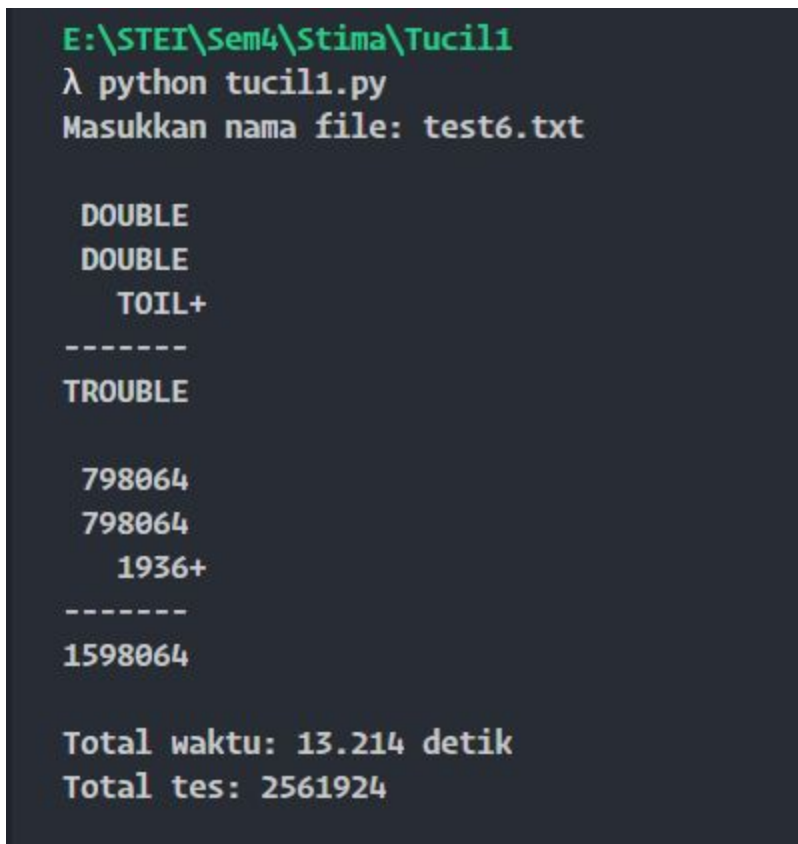
Total waktu: 0.746 detik
Total tes: 203405
```

6. Input



```
test6.txt
1    DOUBLE
2    DOUBLE
3    TOIL+
4    -----
5    TROUBLE
```

Output



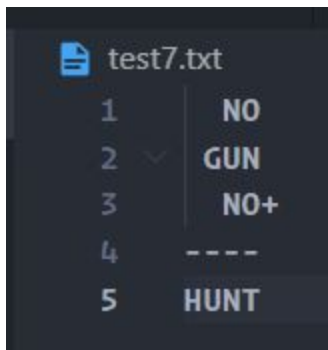
```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test6.txt

DOUBLE
DOUBLE
TOIL+
-----
TROUBLE

798064
798064
1936+
-----
1598064

Total waktu: 13.214 detik
Total tes: 2561924
```

7. Input



1	NO
2	GUN
3	NO+
4	----
5	HUNT

Output

```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test7.txt

    NO
    GUN
    NO+
    ----
    HUNT

    87
    908
    87+
    ----
    1082

Total waktu: 0.093 detik
Total tes: 24858
```

8. Input

```
test8.txt
1  THREE
2  THREE
3   TWO
4   TWO
5   ONE+
6  -----
7  ELEVEN
```

Output

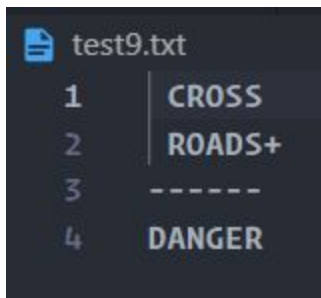
```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test8.txt

THREE
THREE
  TWO
  TWO
  ONE+
-----
ELEVEN

84611
84611
  803
  803
  391+
-----
171219

Total waktu: 7.825 detik
Total tes: 1479027
```

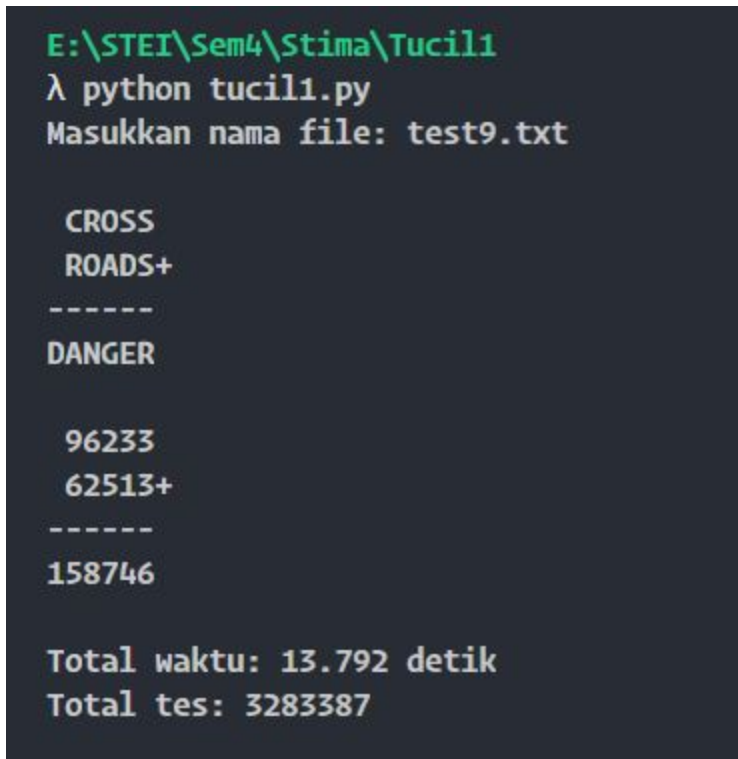
9. Input



A screenshot of a text file named 'test9.txt' with a dark background. The file contains four lines of text, each preceded by a number from 1 to 4. The text is as follows:

No	Input
1	CROSS
2	ROADS+
3	-----
4	DANGER

Output



A screenshot of a terminal window with a dark background. The terminal shows the execution of a Python script named 'tucil1.py'. The output of the script is displayed in a light green color. The output consists of the same four lines of text as the input file, followed by a blank line, and then three more lines of text: '96233', '62513+', and '-----'. The final two lines of the output are 'Total waktu: 13.792 detik' and 'Total tes: 3283387'.

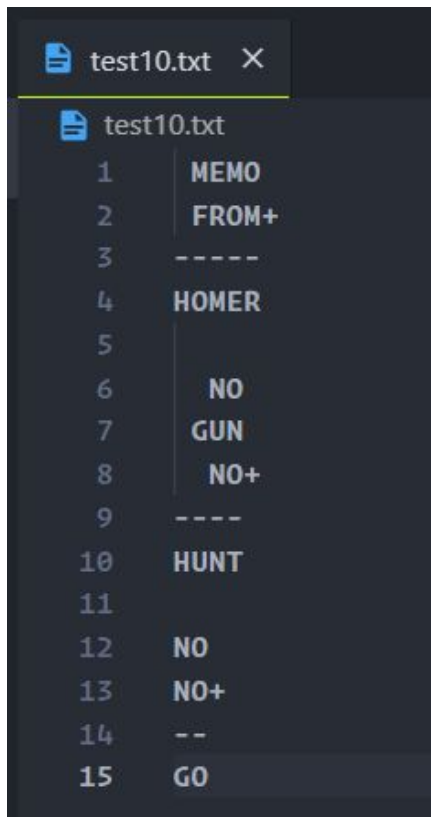
```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test9.txt

CROSS
ROADS+
-----
DANGER

96233
62513+
-----
158746

Total waktu: 13.792 detik
Total tes: 3283387
```


10. Input



```
test10.txt ×
test10.txt
1  MEMO
2  FROM+
3  -----
4  HOMER
5
6  NO
7  GUN
8  NO+
9  ----
10 HUNT
11
12 NO
13 NO+
14 --
15 GO
```

Output

```
E:\STEI\Sem4\Stima\Tucil1
λ python tucil1.py
Masukkan nama file: test10.txt
```

```
MEMO
FROM+
-----
HOMER
```

```
8485
7358+
-----
15843
```

```
Total waktu: 0.432 detik
Total tes: 118227
```

```
NO
GUN
NO+
-----
HUNT
```

```
87
908
87+
-----
1082
```

```
Total waktu: 0.090 detik
Total tes: 25072
```

```
NO
NO+
--
GO
```

```
10
10+
--
20
```

```
Total waktu: 0.001 detik
Total tes: 6
```

Link Alamat Google Drive :

<https://drive.google.com/file/d/1e8h840bhYLIuL6f33-Xoh6uQsGo4x5YI/view?usp=sharing>

Tabel Ceklis :

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	✓	
2. Program berhasil running	✓	
3. Program dapat membaca file masukan dan menuliskan luaran.	✓	
4. Solusi cryptarithmic hanya benar untuk persoalan cryptarithmic dengan dua buah operand		✓
5. Solusi cryptarithmic benar untuk persoalan cryptarithmic untuk lebih dari dua buah operand	✓	