

**Laporan Tugas Kecil 1**  
**Pemecah Persoalan *Cryptarithmic* dengan *Brute Force***

Shifa Salsabiila 13519106

Program Studi Teknik Informatika

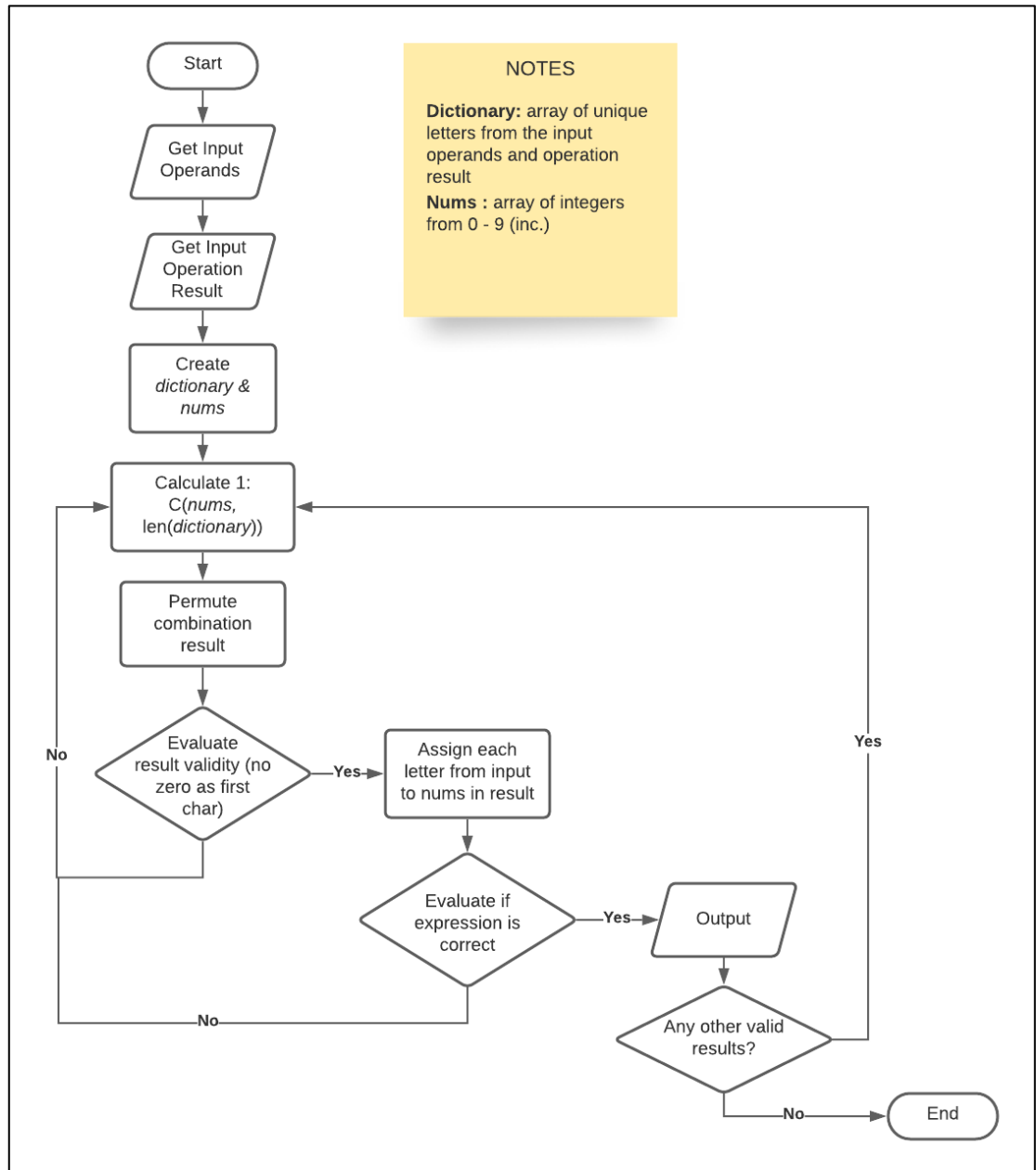
IF2211: Strategi Algoritma

Dr. Nur Ulfa Maulidevi, S.T., M.Sc.

## Daftar Isi

Daftar Isi .....	ii
I.     Algoritma.....	1
II. <i>Source Program</i> .....	3
III.   Hasil dan <i>Test Case</i> .....	6
IV.    Alamat <i>Source Code</i> .....	11

## I. Algoritma



Gambar 1.1 Diagram alur program

Program diawali dengan membaca sebuah file teks yang berisi  $n$  baris, dengan ketentuan:  $n-2$  baris pertama merupakan *operand*, dan *operand* terakhir diakhiri dengan tanda tambah (+). Baris berikutnya merupakan pemisah antara *operand* dan hasil. Baris terakhir adalah hasil penjumlahan. Setelah dibaca, seluruh *operand* akan disimpan dalam sebuah *array* dan hasil akan disimpan dalam sebuah variabel. Kemudian, akan dibuat sebuah *array* baru, yang menyimpan seluruh karakter yang ada pada gabungan *operand* dan hasil tanpa pengulangan, selanjutnya akan

direferensikan sebagai *dictionary*. Selanjutnya, dibuat juga sebuah array yang berisi angka 0 – 9 (inklusif), yang direferensikan sebagai *nums*.

Bagian utama algoritma *brute force* yang digunakan mencakup proses-proses berikut: kombinasi, permutasi, pengubahan kata-kata menjadi bilangan, pengecekan validitas hasil, dan pengecekan kebenaran ekspresi. Pertama, akan dilakukan kombinasi dari *nums*. Akan diperoleh sebanyak  $C(n, r)$  hasil, dengan  $n$ : panjang *array nums* dan  $r$ : panjang *array dictionary*. Kemudian, masing-masing solusi kombinasi akan dimasukkan pada fungsi permutasi. Masing-masing input akan menghasilkan sebanyak  $r!$  hasil permutasi. Berikutnya, masing-masing hasil permutasi akan menjadi basis untuk perumusan transformasi kata-kata pada input menjadi sebuah bilangan.

Selanjutnya, hasil transformasi akan dicek apakah memenuhi syarat bahwa, tidak ada huruf pertama yang merupakan 0. Jika memenuhi, kemudian akan dilakukan pengecekan apakah ketika ekspresi penjumlahan *operand* yang sudah ditransformasi sama dengan hasil yang juga sudah ditransformasi. Jika ya, maka jawaban akan disimpan. Proses utama ini akan diulang hingga tidak ada lagi kemungkinan jawaban lain.

Pada proses ini, perlu diperhatikan bahwa pemanggilan fungsi permutasi, pengubahan kata-kata menjadi bilangan, pengecekan validitas hasil, dan pengecekan kebenaran ekspresi dilakukan di dalam fungsi kombinasi, sehingga program tidak perlu melakukan dua kali proses pengulangan.

## II. Source Program

Bahasa: *Python*

```
def combination(self, initial, n, r, outer_index, inner_index, data, result):
    if (r == 0): #Base case 0: Array of size r achieved
        temp = data.copy()
        x = len(temp)
        self.permutation(temp, 0, x, result)
        return

    elif (n == 0): #Base case 1: Outer index reaches end of initial
        return

    else: #Recursion
        data[inner_index] = initial[outer_index]
        self.combination(initial, n-1, r-1, outer_index+1, inner_index+1,\
            data, result)
        self.combination(initial, n-1, r, outer_index+1, inner_index, data,\
            result)
```

```
def permutation(self, array, first, last, result):
    if first == last: #Base case 0: end of array reached
        temp = array.copy()
        if (self.is_valid(self.operands, self.result, self.letters,\
            temp)):
            result.append(temp)
        return

    else: #Recursion
        for i in range(first, last):
            array[first], array[i] = array[i], array[first]
            self.permutation(array, first+1, last, result)
            array[first], array[i] = array[i], array[first]
```

```
def is_valid(self, list_of_words, res, list_of_letters, list_of_num):
    valid = True
    combined_list = list_of_words.copy()
    combined_list.append(res)
    for word in combined_list:
        idx = list_of_letters.index(word[0]) #Get first letters
        if (list_of_num[idx] == 0): #First letter == 0
            valid = False
    return valid
```

```

def word2num(self, word, list_of_letters, list_of_num):
    num = 0
    for i in range(len(word)):
        idx = list_of_letters.index(word[len(word) - i - 1])
        num += (10**i)*list_of_num[idx]
    return num

```

```

def check(self, list_of_words, res, list_of_num):
    rhs = self.word2num(res, self.letters, list_of_num) #Result
    lhs = 0 #Sum of operands
    for word in list_of_words:
        lhs += self.word2num(word, self.letters, list_of_num)
    return lhs == rhs #Sum of operands == result

```

```

def read_file(self, file_name):
    list_of_words = []
    f = open(file_name, 'r')
    lines = f.readlines()

    for i in range(len(lines) - 2):
        list_of_words.append(lines[i].rstrip())
    list_of_words[-1] = list_of_words[-1][:-1]
    res = lines[-1].rstrip()
    return list_of_words, res

```

```

def get_answers(self):
    #Printing all operands and result
    for i in range(len(self.operands)):
        print("word", i+1, ": ", self.operands[i])
    print("result: " + self.result)
    for p in self.valid_results: #Checking operation correctness
        if (self.check(self.operands, self.result, p)):
            print(p)
            to_be_added = []
            for i in range(len(self.operands)):
                to_be_added.append(str(self.word2num\
                    (self.operands[i], self.letters, p)))
            print(self.word2num(self.operands[i], \
                self.letters, p))
            to_be_added.append(str(self.word2num(self.result,\
                self.letters, p)))
            print(self.word2num(self.result, self.letters, p))
            self.final_results.append(to_be_added)

```

```

def solve(self, operands, result):
    start_time = time.time()

    #Attribute declaration
    self.operands = operands
    self.result = result
    self.valid_results = []
    self.nums = [i for i in range(10)]
    self.final_results = []
    self.solution_number = 0

    #Combining all operands and result as string
    comb_letters = result
    for word in operands:
        comb_letters += word

    #Creating dictionary (array of unique letters)
    self.letters = []
    for letter in comb_letters:
        if letter not in self.letters:
            self.letters.append(letter)

    #Brute force setup
    r = len(self.letters)
    n = len(self.nums)
    data = [0 for i in range(r)]

    self.combination(self.nums, n, r, 0, 0, data, self.valid_results)
    self.get_answers()

    end_time = time.time()
    self.runtime = end_time - start_time
    self.checks = len(self.valid_results)
    print(end_time - start_time)

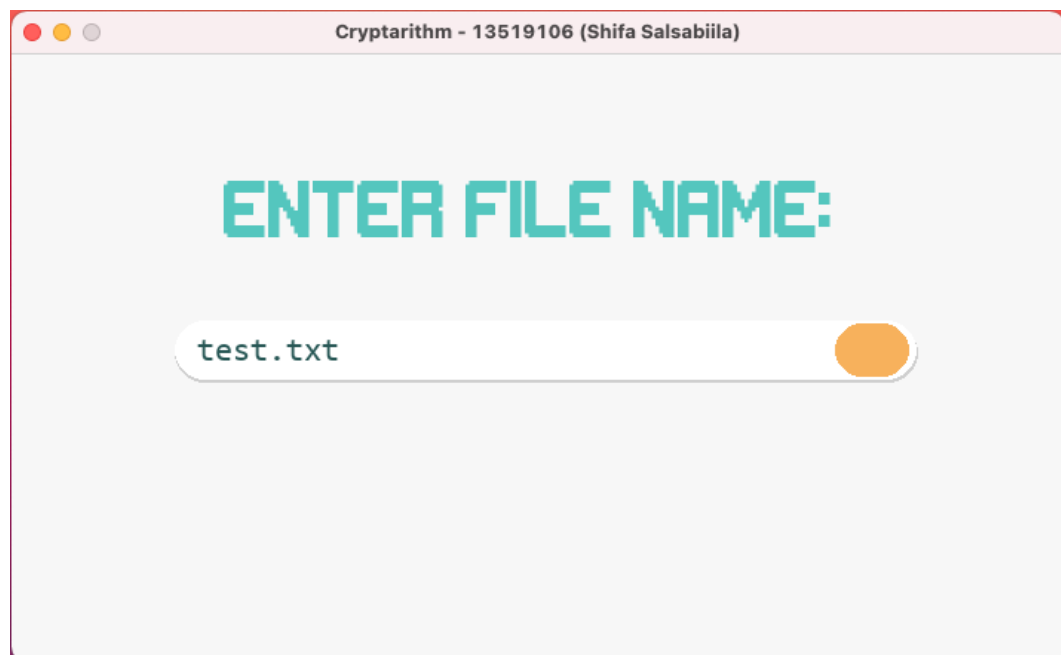
```

Catatan: Hanya *source code* untuk *solver*-nya saja yang dicantumkan pada laporan ini, untuk bagian yang mencakup GUI sengaja tidak diikutsertakan, dengan alasan bahwa algoritma utama penyelesaian masalah hanya terdapat pada bagian ini.

### III. Hasil dan *Test Case*



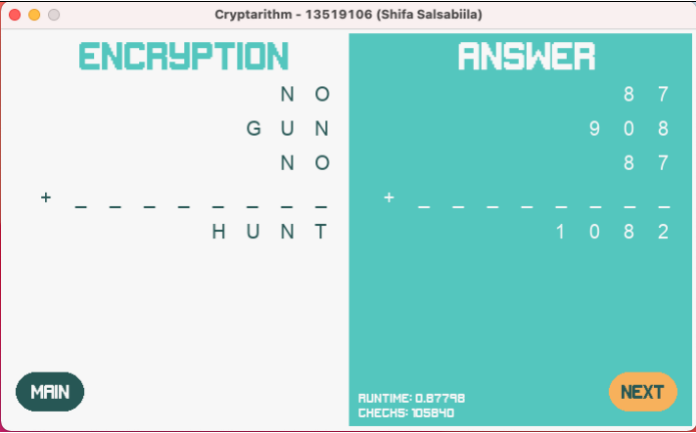
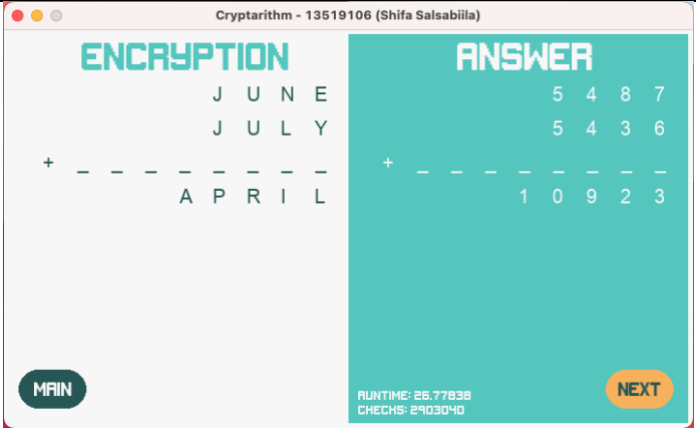
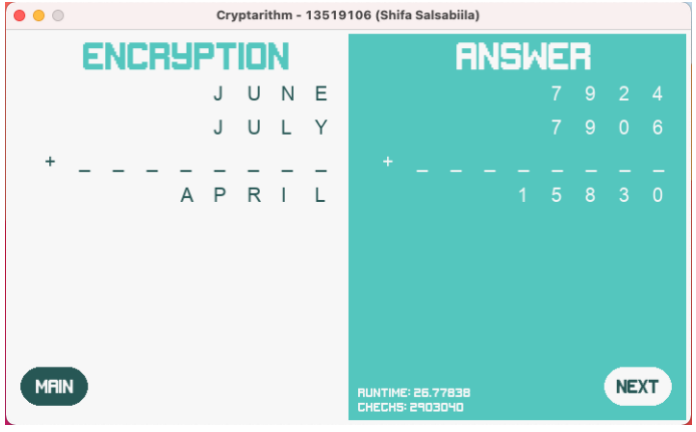
**Gambar 3.1** Tampilan utama program, pilih *auto solve*

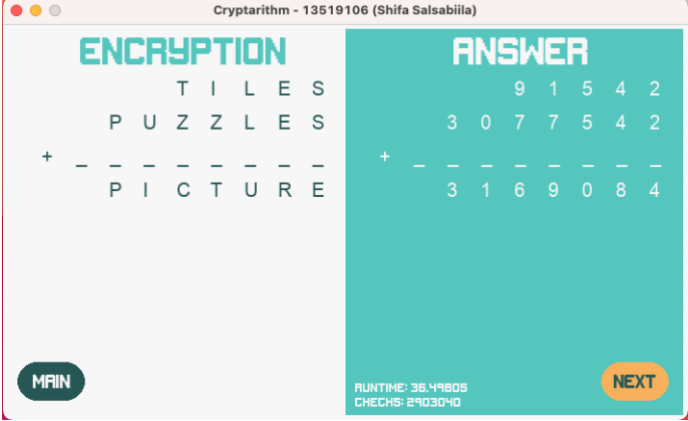
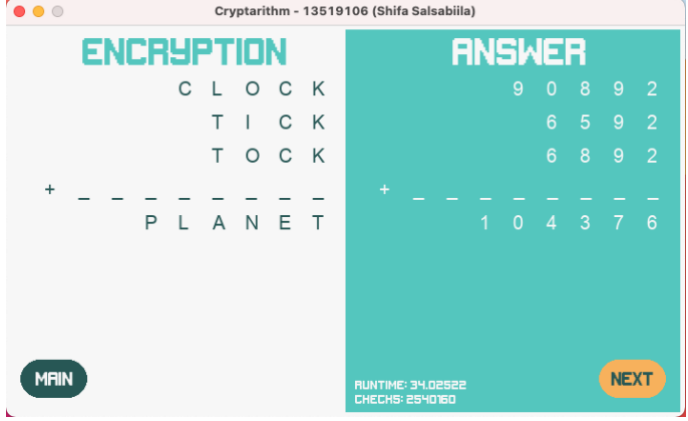
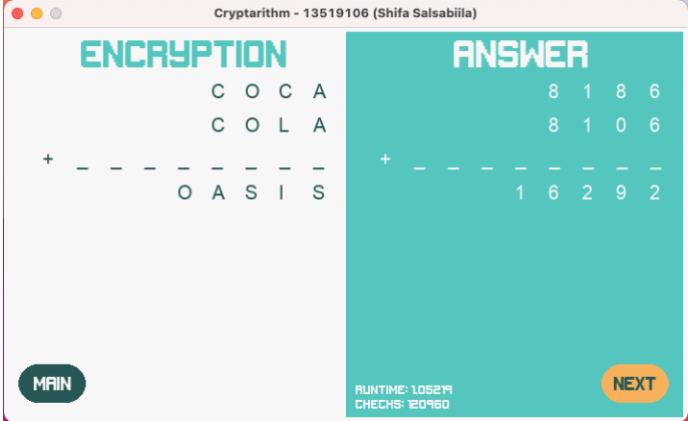


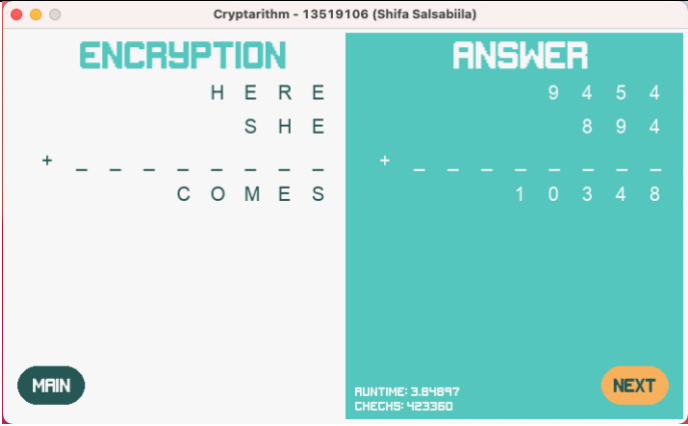
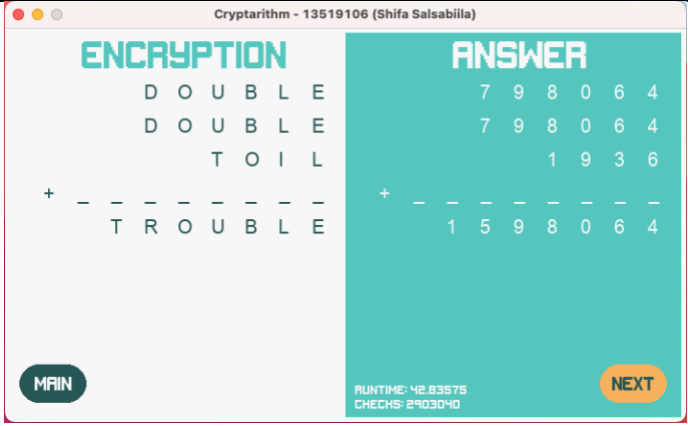

**Gambar 3.2** Tampilan pembacaan nama file

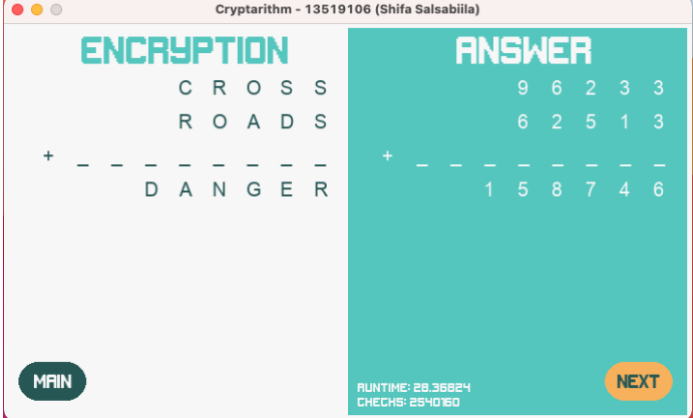




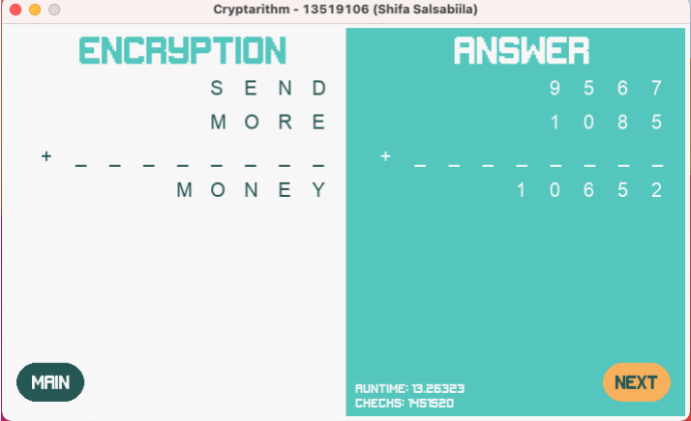
Tabel 3.1 Test case

Input	Output
<div>NO GUN NO+ _____ HUNT</div> <div>File: test.txt</div>	<div><div>Runtime: 0.87798 Checks: 105840</div></div>
<div>JUNE JULY+ _____ APRIL</div> <div>File: test2.txt</div>	<div><div>Runtime: 26.77838 Checks: 2903040</div></div> <div><div>Runtime: 26.77838 Checks: 2903040</div></div>

<p>TILES PUZZLES+ _____ PICTURE</p> <p>File: test3.txt</p>	 <p><b>Runtime: 36.49805</b> <b>Checks: 2903040</b></p>
<p>CLOCK TICK TOCK+ _____ PLANET</p> <p>File: test4.txt</p>	 <p><b>Runtime: 34.02522</b> <b>Checks: 2540160</b></p>
<p>COCA COLA+ _____ OASIS</p> <p>File: test5.txt</p>	 <p><b>Runtime: 1.05219</b> <b>Checks: 120960</b></p>

<p>HERE SHE+ _____ COMES</p> <p>File: test6.txt</p>	 <p><b>Runtime: 3.84897</b> <b>Checks: 423360</b></p>
<p>DOUBLE DOUBLE TOIL+ _____ TROUBLE</p> <p>File: test7.txt</p>	 <p><b>Runtime: 42.83575</b> <b>Checks: 2903040</b></p>
<p>THREE THREE TWO TWO ONE+ _____ ELEVEN</p> <p>File: test8.txt</p>	 <p><b>Runtime: 40.62797</b> <b>Checks: 2540160</b></p>

<p>CROSS ROADS+ ----- DANGER</p> <p>File: test9.txt</p>	 <p>Runtime: 28.36824 Checks: 2540160</p>
<p>MEMO FROM+ ----- HOMER</p> <p>File: test10.txt</p>	 <p>Runtime: 0.93960 Checks: 105840</p>
<p>NUMBER NUMBER+ ----- PUZZLE</p> <p>File: test11.txt</p>	 <p>Runtime: 34.0441 Checks: 2903040</p>

<p>SEND MORE+ MONEY</p> <p>File: test12.txt</p>	 <p>Runtime: 13.26323 Checks: 1451520</p>
---	---

**Tabel 3.2** Form penilaian mandiri

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

#### IV. Alamat Source Code

Repository: <https://github.com/salsabiilashifa11/Cryptarithm>