NAMA   : IRMA ERYANTI PUTRI

NIM   : 1800018272

PRAKTIKUM 7 TEKNIK OPTIMASI KELAS B

File parcel.php

```php
<?php
class Parameters
{
    public $file_name;
    public $indexes;
    public $columns;
    public $population_size;
    public $fitness;
    public $max_generation;
    public $knapsack;
    public $crossover_rate;
    public function __construct($parameters)
    {
        $this->file_name = $parameters['file_name'];
        $this->indexes = $parameters['indexes'];
        $this->columns = $parameters['columns'];
        $this->population_size = $parameters['population_size'];
        $this->fitness = $parameters['fitness'];
        $this->max_generation = $parameters['max_generation'];
        $this->knapsack = $parameters['knapsack'];
        $this->crossover_rate = $parameters['crossover_rate'];
    }
}
class Products
{
    public static function catalogue($parameters)
    {
        ## Todo change into open closed principles/polymorphism
        if (!is_array($parameters->indexes)) {
            return new Exception("Indexes parameter must be an array!");
        }
        if (!is_array($parameters->columns)) {
            return new Exception("Columns parameter must be an array!");
        }
        if (!is_string($parameters->file_name)) {
```

```php
        }
        if (!is_string($parameters->file_name)) {
            return new Exception("Filename parameter must be a string!");
        }
        $raw_data = file($parameters->file_name);
        foreach ($raw_data as $val) {
            $data[] = explode(",", $val);
        }
        foreach ($data as $key => $val) {
            foreach (array_keys($val) as $subkey) {
                if ($subkey == $parameters->indexes[$subkey]) {
                    $data[$key][$parameters->columns[$subkey]] = $data[$key][$subkey];
                    unset($data[$key][$subkey]);
                }
            }
        }
        return [
            'dataset' => $data,
            'gen_length' => count($data)
        ];
    }
}
class Generate
{
    public static function initialPopulation($parameters)
    {
        $catalogue = Products::catalogue($parameters);
        for ($i = 0; $i <= $parameters->population_size - 1; $i++) {
            for ($j = 0; $j <= $catalogue['gen_length'] - 1; $j++) {
                $ret[$i][$j] = rand(0, 1);
            }
        }
        return $ret;
    }
}
```

```php
 69        public static function randomZeroToOne()
 70        {
 71            return (float) rand() / (float) getrandmax();
 72        }
 73
 74        public static function randomGenLength($parameters)
 75        {
 76            $catalogue = Products::catalogue($parameters);
 77            return rand(1, $catalogue['gen_length'] - 1);
 78        }
 79    }
 80    class Pairing
 81    {
 82        public static function chromosomeGenWithProducts($populations, $parameters)
 83        {
 84            $ret = [];
 85            echo "Populations: <br>";
 86            foreach ($populations as $no => $solution) {
 87                echo $no . ' ';
 88                foreach ($solution as $key => $gen) {
 89                    print_r($gen);
 90                }
 91                echo '<br>';
 92            }
 93            echo '<p>';
 94            foreach ($populations as $no => $solution) {
 95                foreach ($solution as $key => $gen) {
 96                    if ($gen === 1) {
 97                        $selected_product = Products::catalogue($parameters)['dataset'][$key]['item'];
 98                        $price = Products::catalogue($parameters)['dataset'][$key]['price'];
 99                        $ret[$no][] = [
100                            'product' => $selected_product,
101                            'price' => $price
102                        ];
103                    }
```

Spaces: 4   PHP

```php
106            return $ret;
107        }
108    }
109    class Calculate
110    {
111        public static function sumOfBuy($selected_product)
112        {
113            $ret = [];
114            foreach ($selected_product as $key => $products) {
115                echo $key . '<br>';
116                print_r($products);
117                //$sum = array_sum(array_column($products, 'price'));
118                //echo $sum;
119                //$ret[] = $sum;
120                echo '<br>';
121            }
122            return $ret;
123        }
124    }
125    class Fitness
126    {
127        public static function evaluation($sumOfBuy, $knapsack)
128        {
129            $ret = [];
130            $negative_residu = 2;
131            $positive_residu = 1;
132            foreach ($sumOfBuy as $total) {
133                $residu = $knapsack - $total;
134                if ($residu < 0) {
135                    $ret[] = 1 / (1 + $negative_residu);
136                }
137                if ($residu > 0) {
138                    $ret[] = 1 / (1 + $positive_residu);
139                }
140                if ($residu === 0) {
```

Spaces: 4   PHP

```php
140                if ($residu === 0) {
141                    $ret[] = 1;
142                }
143            }
144            return $ret;
145        }
146        public static function isOne($individual_fitness)
147        {
148            if (array_search(1, $individual_fitness)) {
149                return true;
150            }
151        }
152        public static function result($population, $individual_fitness, $sumOfBuy)
153        {
154            $max_value = 1 / 2;
155            foreach ($individual_fitness as $key => $fitness_value) {
156                if ($max_value === $fitness_value) {
157                    $indexes[] = $key;
158                }
159            }
160            foreach ($sumOfBuy as $key => $val) {
161                foreach ($indexes as $subyek => $ind) {
162                    if ($key === $ind) {
163                        $optimized[] = [
164                            'index' => $key,
165                            'price' => $val
166                        ];
167                        break;
168                    }
169                }
170            }
171            $price = max(array_column($optimized, 'price'));
172            $id = array_search($price, array_column($optimized, 'price'));
173            $individu = $population[$optimized[$id]['index']];
174            return [
```

```php
174            return [
175                'price' => $price,
176                'solution' => $individu
177            ];
178        }
179    }
180    class RandomNumbers
181    {
182        public static function zeroToOne($population_size)
183        {
184            for ($i = 0; $i <= $population_size - 1; $i++) {
185                $ret[] = Generate::randomZeroToOne();
186            }
187            return $ret;
188        }
189    }
190    class RouletteWheelSelection
191    {
192        public $population_size;
193
194        function __construct($population_size)
195        {
196            $this->population_size = $population_size;
197        }
198        function probability($individual_fitness)
199        {
200            foreach ($individual_fitness as $fitness) {
201                $ret[] = $fitness / array_sum($individual_fitness);
202            }
203            return $ret;
204        }
205        function cummulative($individual_fitness)
206        {
207            foreach ($this->probability($individual_fitness) as $key => $probability) {
208                if ($key === 0) {
```

```php
        foreach ($this->probability($individual_fitness) as $key => $probability) {
            if ($key === 0) {
                $ret[$key] = $probability;
            } else {
                $ret[$key] = $probability + $ret[$key - 1];
            }
        }
        return $ret;
    }
    function selection($individual_fitness, $initial_populations)
    {
        $randomZeroToOne = RandomNumbers::zeroToOne($this->population_size);
        foreach ($randomZeroToOne as $key => $random) {
            echo $random;
            echo '<br>';
            foreach ($this->cummulative($individual_fitness) as $subkey => $roulette) {
                echo $roulette . ' ';
                if ($random < $roulette) {
                    echo ' [ ' . $subkey . ' ' . $roulette . ' ] ';
                    $ret[$key] = $initial_populations[$subkey];
                    break;
                }
            }
            echo '<br>';
        }
        return $ret;
    }
}
class CrossOver
{
    public $population_size;
    public $crossover_rate;

    function __construct($population_size, $crossover_rate)
    {
```

```php
        $this->population_size = $population_size;
        $this->crossover_rate = $crossover_rate;
    }
    function selectedIndividual()
    {
        $randomZeroToOne = RandomNumbers::zeroToOne($this->population_size);
        foreach ($randomZeroToOne as $key => $val) {
            if ($val < $this->crossover_rate) {
                $ret[] = $key;
            }
        }
        return $ret;
    }

    function generateCombination($selectedIndividual)
    {
        foreach ($selectedIndividual as $val) {
            $acak = $selectedIndividual[array_rand($selectedIndividual)];
            $result[] = [$val, $acak];
        }
        return $result;
    }

    function combination($selectedIndividual)
    {
        $counter = 0;
        while ($counter < 1) {
            $combination = $this->generateCombination($selectedIndividual);
            foreach ($combination  as $val) {
                $sum[] = count(array_unique($val));
            }
            if (array_sum($sum) === count($combination) * 2) {
                return $combination;
            }
            $counter = 0;
```

```php
276            $counter = 0;
277            $sum = [];
278        }
279    }
280
281    function cutPositions($parameters, $number_of_parents)
282    {
283        for ($i = 0; $i <= $number_of_parents - 1; $i++) {
284            $ret[] = Generate::randomGenLength($parameters);
285        }
286        return $ret;
287    }
288
289    function newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters)
290    {
291        $catalogue = Products::catalogue($parameters);
292        $offspring = [];
293
294        foreach ($combinations as $key => $combination) {
295            $individu1 = $selected_chromosome[$combination[0]];
296            $individu2 = $selected_chromosome[$combination[1]];
297
298            for ($i = 0; $i <= $catalogue['gen_length'] - 1; $i++) {
299                if ($i < $cut_positions[$key]) {
300                    $offspring[$key][] = $individu1[$i];
301                } else {
302                    $offspring[$key][] = $individu2[$i];
303                }
304            }
305            $ret[] = [
306                'index' => $selected_individuals[$key],
307                'offspring' => $offspring[$key]
308            ];
309        }
310        return $ret;
```

```php
312
313    function updatePopulations($selected_chromosome, $new_individuals)
314    {
315        for ($i = 0; $i <= count($selected_chromosome) - 1; $i++) {
316            foreach ($new_individuals as $key => $individual) {
317                if ($i === $individual['index']) {
318                    $ret[$i][$key] = [
319                        'a' => 'new',
320                        'b' => $i
321                    ];
322                } else {
323                    $ret[$i][$key] = [
324                        'a' => 'old',
325                        'b' => $i
326                    ];
327                }
328            }
329        }
330
331        foreach ($ret as $key => $z) {
332            $index = array_search('new', array_column($z, 'a'));
333            if ($index || $index === 0) {
334                $index_new = array_search($z[$index]['b'], array_column($new_individuals, 'index'));
335                $result[] = $new_individuals[$index_new]['offspring'];
336            } else {
337                $result[] = $selected_chromosome[$key];
338            }
339        }
340        return $result;
341    }
342 }
343
344 $parameters = [
345     'file_name' => 'products.txt',
346     'indexes' => [0, 1],
```

```php
344  $parameters = [
345      'file_name' => 'products.txt',
346      'indexes' => [0, 1],
347      'columns' => ['item', 'price'],
348      'population_size' => 10,
349      'fitness' => 1000,
350      'max_generation' => 1000,
351      'crossover_rate' => 0.75,
352      'knapsack' => 150000
353  ];
354
355  $parameters = new Parameters($parameters);
356  $catalogue = Products::catalogue($parameters);
357
358  for ($i = 0; $i <= $parameters->max_generation; $i++) {
359      if ($i === 0) {
360          $initial_populations = Generate::initialPopulation($parameters);
361          $selected_product = Pairing::chromosomeGenWithProducts($initial_populations, $parameters);
362          echo '<p>';
363          $sumOfBuy = Calculate::sumOfBuy($selected_product);
364          print_r($sumOfBuy);
365          echo '<p>';
366          $individual_fitness = Fitness::evaluation($sumOfBuy, $parameters->knapsack);
367          print_r($individual_fitness);
368
369          echo ' Total fitness ' . array_sum($individual_fitness);
370          echo '<br>';
371
372          if (Fitness::isOne($individual_fitness)) {
373              echo ' Total fitness ' . array_sum($individual_fitness);
374              echo 'Maximum 1 achieved!';
375              $solution = Fitness::result($initial_populations, $individual_fitness, $sumOfBuy);
376              print_r($solution);
377              exit();
378          }
```

```php
378      }
379          $solution = Fitness::result($initial_populations, $individual_fitness, $sumOfBuy);
380          print_r($solution);
381          $bests[] = $solution;
382
383          $select = new RouletteWheelSelection($parameters->population_size);
384          echo '<p>';
385          print_r($select->probability($individual_fitness));
386          echo '<p>';
387          $cummulative = $select->cummulative($individual_fitness);
388          echo '<p>';
389          print_r($cummulative);
390
391          $selected_chromosome = $select->selection($individual_fitness, $initial_populations);
392          echo '<p>';
393          echo 'Hasil seleksi<br>';
394          print_r($selected_chromosome);
395          echo '<br>';
396          $crossover = new CrossOver($parameters->population_size, $parameters->crossover_rate);
397          $selected_individuals = $crossover->selectedIndividual($selected_chromosome, $initial_populations);
398          $number_of_parents = count($selected_individuals);
399          echo 'Individu terpilih: ';
400          echo $number_of_parents;
401          echo ' ';
402          print_r($selected_individuals);
403          echo '<p>Combination:<br>';
404          $combinations = $crossover->combination($selected_individuals);
405          print_r($combinations);
406          echo '<p>';
407          $cut_positions = $crossover->cutPositions($parameters, $number_of_parents);
408          echo 'Cut position: ';
409          print_r($cut_positions);
410          echo '<br>';
411          $new_individuals = $crossover->newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters);
412          print_r($new_individuals);
```

```php
412        print_r($new_individuals);
413
414        echo '<p>';
415        echo 'New populations<br>';
416        $new_populations = $crossover->updatePopulations($selected_chromosome, $new_individuals);
417        print_r($new_populations);
418        echo '<p>';
419    }
420
421    if ($i > 0) {
422        echo ' Generasi: ' . $i . '<br>';
423        $selected_product = Pairing::chromosomeGenWithProducts($new_populations, $parameters);
424        echo '<p>';
425        $sumOfBuy = Calculate::sumOfBuy($selected_product);
426        print_r($sumOfBuy);
427        echo '<p>';
428        $individual_fitness = Fitness::evaluation($sumOfBuy, $parameters->knapsack);
429        print_r($individual_fitness);
430
431        echo ' Total fitness ' . array_sum($individual_fitness);
432        echo '<br>';
433
434        if (Fitness::isOne($individual_fitness)) {
435            echo ' Total fitness ' . array_sum($individual_fitness);
436            echo 'Maximum 1 achieved!';
437            $solution = Fitness::result($new_populations, $individual_fitness, $sumOfBuy);
438            print_r($solution);
439            exit();
440        }
441        $solution = Fitness::result($new_populations, $individual_fitness, $sumOfBuy);
442        print_r($solution);
443        $bests[] = $solution;
444
445        $select = new RouletteWheelSelection($parameters->population_size);
446
```

Spaces: 4   PHP

```php
444
445        $select = new RouletteWheelSelection($parameters->population_size);
446        echo '<p>';
447        print_r($select->probability($individual_fitness));
448        echo '<p>';
449        $cummulative = $select->cummulative($individual_fitness);
450        echo '<p>';
451        print_r($cummulative);
452
453        $selected_chromosome = $select->selection($individual_fitness, $initial_populations);
454        echo '<p>';
455        echo 'Hasil seleksi<br>';
456        print_r($selected_chromosome);
457        echo '<br>';
458        $crossover = new CrossOver($parameters->population_size, $parameters->crossover_rate);
459        $selected_individuals = $crossover->selectedIndividual($selected_chromosome, $new_populations);
460        $number_of_parents = count($selected_individuals);
461        echo 'Individu terpilih: ';
462        echo $number_of_parents;
463        echo ' ';
464        print_r($selected_individuals);
465        echo '<p>Combination:<br>';
466        $combinations = $crossover->combination($selected_individuals);
467        print_r($combinations);
468        echo '<p>';
469        $cut_positions = $crossover->cutPositions($parameters, $number_of_parents);
470        echo 'Cut position: ';
471        print_r($cut_positions);
472        echo '<br>';
473        $new_individuals = $crossover->newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters);
474        print_r($new_individuals);
475
476        echo '<p>';
477        echo 'New populations<br>';
```

Spaces: 4   PHP

```php
         echo '<p>';
         echo 'New populations<br>';
         $new_populations = $crossover->updatePopulations($selected_chromosome, $new_individuals);
         print_r($new_populations);
         echo '<p>';
     }
 }
 echo '<p>';
 echo '<p>Final: <br>';
 print_r($bests);
 echo '<br>';
 $max_price = max(array_column($bests, 'price'));
 $index = array_search($max_price, array_column($bests, 'price'));
 echo '<p>';
 print_r($bests[$index]);
 $selected_product = Pairing::chromosomeGenWithProducts($bests[$index], $parameters);
```

File products.txt

```
1   Chitato 68 gr, 8900
2   Teh Sosro Kotak, 6900
3   Botan Mackarel 425 gr, 28900
4   Keju Cheddar 180 gr, 15250
5   Palm Fruit Kurma 500 gr, 56900
6   Marjan Syrup 460 ml, 17900
7   365 Wafer Stick 500 gr, 32390
8   Nissin Biscuit Lemonia Twist 340 gr, 22500
9   Kokola Wafer Cream 252 gr, 18500
10  Sari Kacang Hijau 150 ml, 7790
11  Pop Mie Pake Nasi 75 gr, 8000
12  Teriyaki Saori 275 ml, 17900
13  Dua Belibis Sambal 135/340/535 gr, 8650
14  Teh Hijau Kepala Djenggot 60 gr, 11900
15  Madurasa 150 gr, 16900
16  Makaroniku 200 gr, 14900
17  Ultra Low Fat Susu UHT 1000 ml, 19900
18  Khong Guan Malkist Salut Cokelat 120 gr, 4990
19  Kopi susu ABC, 3000
20  Coca cola, 15000
21  Khong guan, 113000
22  Danis biscuit, 46900
23
```

Output:

```
Populations:
0 101001011101000101010011
1 110100110100000110010
2 00011011000000101010100
3 110100100011000001011
4 0010100000010001010101
5 100000110010011100001
6 101101001010000100101
7 011110100001100100001
8 100010001110101100011
9 10001010011110011000111
```

0
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => Botan Mackarel 425 gr [price] => 28900) [2] => Array ([product] => Marjan Syrup 460 ml [price] => 17900) [3] => Array ([product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500) [4] => Array ([product] => Kokola Wafer Cream 252 gr [price] => 18500) [5] => Array ([product] => Sari Kacang Hijau 150 ml [price] => 7790) [6] => Array ([product] => Teriyaki Saori 275 ml [price] => 17900) [7] => Array ([product] => Makaroniku 200 gr [price] => 14900) [8] => Array ([product] => Khong Guan Malkist Salut Cokelat 120 gr [price] => 4990) [9] => Array ([product] => Khong guan [price] => 113000) [10] => Array ([product] => Danis biscuit [price] => 46900))

1
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => Teh Sosro Kotak [price] => 6900) [2] => Array ([product] => Keju Cheddar 180 gr [price] => 15250) [3] => Array ([product] => 365 Wafer Stick 500 gr [price] => 32390) [4] => Array ([product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500) [5] => Array ([product] => Sari Kacang Hijau 150 ml [price] => 7790) [6] => Array ([product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900) [7] => Array ([product] => Khong Guan Malkist Salut Cokelat 120 gr [price] => 4990) [8] => Array ([product] => Khong guan [price] => 113000))

2
Array ([0] => Array ([product] => Keju Cheddar 180 gr [price] => 15250) [1] => Array ([product] => Palm Fruit Kurma 500 gr [price] => 56900) [2] => Array ([product] => 365 Wafer Stick 500 gr [price] => 32390) [3] => Array ([product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500) [4] => Array ([product] => Madurasa 150 gr [price] => 16900) [5] => Array ([product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900) [6] => Array ([product] => Khong Guan Malkist Salut Cokelat 120 gr [price] => 4990) [7] => Array ([product] => Coca cola [price] => 15000))

3
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => Teh Sosro Kotak [price] => 6900) [2] => Array ([product] => Keju Cheddar 180 gr [price] => 15250) [3] => Array ([product] => 365 Wafer Stick 500 gr [price] => 32390) [4] => Array ([product] => Pop Mie Pake Nasi 75 gr [price] => 8000) [5] => Array ([product] => Teriyaki Saori 275 ml [price] => 17900) [6] => Array ([product] => Kopi susu ABC [price] => 3000) [7] => Array ([product] => Khong guan [price] => 113000) [8] => Array ([product] => Danis biscuit [price] => 46900))

4
Array ([0] => Array ([product] => Botan Mackarel 425 gr [price] => 28900) [1] => Array ([product] => Palm Fruit Kurma 500 gr [price] => 56900) [2] => Array ([product] => Teriyaki Saori 275 ml [price] => 17900) [3] => Array ([product] => Makaroniku 200 gr [price] => 14900) [4] => Array ([product] => Khong Guan Malkist Salut Cokelat 120 gr [price] => 4990) [5] => Array ([product] => Coca cola [price] => 15000) [6] => Array ([product] => Danis biscuit [price] => 46900))

5

5
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => 365 Wafer Stick 500 gr [price] => 32390) [2] => Array ([product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500) [3] => Array ([product] => Pop Mie Pake Nasi 75 gr [price] => 8000) [4] => Array ([product] => Teh Hijau Kepala Djenggot 60 gr [price] => 11900) [5] => Array ([product] => Madurasa 150 gr [price] => 16900) [6] => Array ([product] => Makaroniku 200 gr [price] => 14900) [7] => Array ([product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900) [8] => Array ([product] => Danis biscuit [price] => 46900))

6
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => Botan Mackarel 425 gr [price] => 28900) [2] => Array ([product] => Keju Cheddar 180 gr [price] => 15250) [3] => Array ([product] => Marjan Syrup 460 ml [price] => 17900) [4] => Array ([product] => Kokola Wafer Cream 252 gr [price] => 18500) [5] => Array ([product] => Pop Mie Pake Nasi 75 gr [price] => 8000) [6] => Array ([product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900) [7] => Array ([product] => Coca cola [price] => 15000) [8] => Array ([product] => Danis biscuit [price] => 46900))

7
Array ([0] => Array ([product] => Teh Sosro Kotak [price] => 6900) [1] => Array ([product] => Botan Mackarel 425 gr [price] => 28900) [2] => Array ([product] => Keju Cheddar 180 gr [price] => 15250) [3] => Array ([product] => Marjan Syrup 460 ml [price] => 17900) [4] => Array ([product] => Teriyaki Saori 275 ml [price] => 17900) [5] => Array ([product] => Dua Belibis Sambal 135/340/535 gr [price] => 8650) [6] => Array ([product] => Makaroniku 200 gr [price] => 14900) [7] => Array ([product] => Danis biscuit [price] => 46900))

8
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => Palm Fruit Kurma 500 gr [price] => 56900) [2] => Array ([product] => Kokola Wafer Cream 252 gr [price] => 18500) [3] => Array ([product] => Sari Kacang Hijau 150 ml [price] => 7790) [4] => Array ([product] => Pop Mie Pake Nasi 75 gr [price] => 8000) [5] => Array ([product] => Dua Belibis Sambal 135/340/535 gr [price] => 8650) [6] => Array ([product] => Madurasa 150 gr [price] => 16900) [7] => Array ([product] => Makaroniku 200 gr [price] => 14900) [8] => Array ([product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900) [9] => Array ([product] => Khong guan [price] => 113000) [10] => Array ([product] => Danis biscuit [price] => 46900))

9
Array ([0] => Array ([product] => Chitato 68 gr [price] => 8900) [1] => Array ([product] => Keju Cheddar 180 gr [price] => 15250) [2] => Array ([product] => Marjan Syrup 460 ml [price] => 17900) [3] => Array ([product] => Kokola Wafer Cream 252 gr [price] => 18500) [4] => Array ([product] => Sari Kacang Hijau 150 ml [price] => 7790) [5] => Array ([product] => Pop Mie Pake Nasi 75 gr [price] => 8000) [6] => Array ([product] => Teriyaki Saori 275 ml [price] => 17900) [7] => Array ([product] => Madurasa 150 gr [price] => 16900) [8] => Array ([product] => Makaroniku 200 gr [price] => 14900) [9] => Array ([product] => Coca cola [price] => 15000) [10] => Array ([product] => Khong guan [price] => 113000) [11] => Array ([product] => Danis biscuit [price] => 46900))
Array ( )

Array ( ) Total fitness 0

Penjelasan Singkat :

File **parcel.php** yang sudah didownload di github dan dirunning menggunakan dataset **products.txt** Sehingga didapatkan hasil output seperti gambar output diatas, untuk hasil pengerjaanya dapat dilihat dari gambar-gambar diatas.