

```
parcel.php X
parcel.php > ...
1  <?php
2
3  class Parameters
4  {
5      public $file_name;
6      public $indexes;
7      public $columns;
8      public $population_size;
9      public $fitness;
10     public $max_generation;
11     public $knapsack;
12     public $crossover_rate;
13
14     public function __construct($parameters)
15     {
16         $this->file_name = $parameters['file_name'];
17         $this->indexes = $parameters['indexes'];
18         $this->columns = $parameters['columns'];
19         $this->population_size = $parameters['population_size'];
20         $this->fitness = $parameters['fitness'];
21         $this->max_generation = $parameters['max_generation'];
22         $this->knapsack = $parameters['knapsack'];
23         $this->crossover_rate = $parameters['crossover_rate'];
24     }
25 }
26
27 class Products
28 {
29     public static function catalogue($parameters)
30     {
31         ## Todo change into open closed principles/polymorphism
32         if (!is_array($parameters->indexes)) {
33             return new Exception("Indexes parameter must be an array!");
34         }
35         if (!is_array($parameters->columns)) {
36             return new Exception("Columns parameter must be an array!");
37         }
38         if (!is_string($parameters->file_name)) {
39             return new Exception("Filename parameter must be a string!");
40         }
41
42         $raw_data = file($parameters->file_name);
43         foreach ($raw_data as $val) {
44             $data[] = explode(",", $val);
45         }
46
47         foreach ($data as $key => $val) {
48             foreach (array_keys($val) as $subkey) {
49                 if ($subkey == $parameters->indexes[$subkey]) {
50                     $data[$key][$parameters->columns[$subkey]] = $data[$key][$subkey];
51                     unset($data[$key][$subkey]);
52                 }
53             }
54         }
55     }
56 }
```

```
Explorer (Ctrl+Shift+E)
parcel.php > ...

42     $raw_data = file($parameters->file_name);
43     foreach ($raw_data as $val) {
44         $data[] = explode(",", $val);
45     }
46
47     foreach ($data as $key => $val) {
48         foreach (array_keys($val) as $subkey) {
49             if ($subkey == $parameters->indexes[$subkey]) {
50                 $data[$key][$parameters->columns[$subkey]] = $data[$key][$subkey];
51                 unset($data[$key][$subkey]);
52             }
53         }
54     }
55     return [
56         'dataset' => $data,
57         'gen_length' => count($data)
58     ];
59 }
60
61
62 class Generate
63 {
64     public static function initialPopulation($parameters)
65     {
66         $catalogue = Products::catalogue($parameters);
67         for ($i = 0; $i <= $parameters->population_size - 1; $i++) {
68             for ($j = 0; $j <= $catalogue['gen_length'] - 1; $j++) {
69                 $ret[$i][$j] = rand(0, 1);
70             }
71         }
72         return $ret;
73     }
74
75     public static function randomZeroToOne()
76     {
77         return (float) rand() / (float) getrandmax();
78     }
79
80     public static function randomGenLength($parameters)
81     {
82         $catalogue = Products::catalogue($parameters);
83         return rand(1, $catalogue['gen_length'] - 1);
84     }
85 }
86
87 class Pairing
88 {
89     public static function chromosomeGenWithProducts($populations, $parameters)
90     {
91         $ret = [];
92         echo "Populations: <br>";
```

```
parcel.php X
parcel.php > ...
Search (Ctrl+Shift+F)
87 class Pairing
88 {
89     public static function chromosomeGenWithProducts($populations, $parameters)
90     {
91         $ret = [];
92         echo "Populations: <br>";
93         foreach ($populations as $no => $solution) {
94             echo $no . ' ' ;
95             foreach ($solution as $key => $gen) {
96                 print_r($gen);
97             }
98             echo '<br>';
99         }
100         echo '<p>';
101         foreach ($populations as $no => $solution) {
102             foreach ($solution as $key => $gen) {
103                 if ($gen == 1) {
104                     $selected_product = Products::catalogue($parameters)['dataset'][$key]['item'];
105                     $price = Products::catalogue($parameters)['dataset'][$key]['price'];
106                     $ret[$no][] = [
107                         'product' => $selected_product,
108                         'price' => $price
109                     ];
110                 }
111             }
112         }
113         return $ret;
114     }
115 }
116
117 class Calculate
118 {
119     public static function sumOfBuy($selected_product)
120     {
121         $ret = [];
122         foreach ($selected_product as $key => $products) {
123             echo $key . '<br>';
124             print_r($products);
125             $sum = array_sum(array_column($products, 'price'));
126             echo $sum;
127             $ret[] = $sum;
128             echo '<br>';
129         }
130         return $ret;
131     }
132 }
133
134 class Fitness
135 {
```

```
Explorer (Ctrl+Shift+E)
parcel.php > ...
133
134 class Fitness
135 {
136     public static function evaluation($sumOfBuy, $knapsack)
137     {
138         $ret = [];
139         $negative_residu = 2;
140         $positive_residu = 1;
141         foreach ($sumOfBuy as $total) {
142             $residu = $knapsack - $total;
143             if ($residu < 0) {
144                 $ret[] = 1 / (1 + $negative_residu);
145             }
146             if ($residu > 0) {
147                 $ret[] = 1 / (1 + $positive_residu);
148             }
149             if ($residu === 0) {
150                 $ret[] = 1;
151             }
152         }
153         return $ret;
154     }
155
156     public static function isOne($individual_fitness)
157     {
158         if (array_search(1, $individual_fitness)) {
159             return true;
160         }
161     }
162
163     public static function result($population, $individual_fitness, $sumOfBuy)
164     {
165         $max_value = 1 / 2;
166         foreach ($individual_fitness as $key => $fitness_value) {
167             if ($max_value === $fitness_value) {
168                 $indexes[] = $key;
169             }
170         }
171         foreach ($sumOfBuy as $key => $val) {
172             foreach ($indexes as $subyek => $ind) {
173                 if ($key === $ind) {
174                     $optimized[] = [
175                         'index' => $key,
176                         'price' => $val
177                     ];
178                     break;
179                 }
180             }
181         }
182         $price = max(array_column($optimized, 'price'));
183         $id = array_search($price, array_column($optimized, 'price'));
```

```
parcel.php x
parcel.php > ...

181     }
182     $price = max(array_column($optimized, 'price'));
183     $id = array_search($price, array_column($optimized, 'price'));
184     $individu = $population[$optimized[$id]['index']];
185     return [
186         'price' => $price,
187         'solution' => $individu
188     ];
189 }
190 }
191
192 class RandomNumbers
193 {
194     public static function zeroToOne($population_size)
195     {
196         for ($i = 0; $i <= $population_size - 1; $i++) {
197             $ret[] = Generate::randomZeroToOne();
198         }
199         return $ret;
200     }
201 }
202
203 class RouletteWheelSelection
204 {
205     public $population_size;
206
207     function __construct($population_size)
208     {
209         $this->population_size = $population_size;
210     }
211
212     function probability($individual_fitness)
213     {
214         foreach ($individual_fitness as $fitness) {
215             $ret[] = $fitness / array_sum($individual_fitness);
216         }
217         return $ret;
218     }
219
220     function cummulative($individual_fitness)
221     {
222         foreach ($this->probability($individual_fitness) as $key => $probability) {
223             if ($key === 0) {
224                 $ret[$key] = $probability;
225             } else {
226                 $ret[$key] = $probability + $ret[$key - 1];
227             }
228         }
229         return $ret;
230     }
231 }
```

```
parcel.php x
parcel.php > ...
223     if ($key == 0) {
224         $ret[$key] = $probability;
225     } else {
226         $ret[$key] = $probability + $ret[$key - 1];
227     }
228 }
229 return $ret;
230 }
231
232 function selection($individual_fitness, $initial_populations)
233 {
234     $randomZeroToOne = RandomNumbers::zeroToOne($this->population_size);
235     foreach ($randomZeroToOne as $key => $random) {
236         echo $random;
237         echo '<br>';
238         foreach ($this->cummulative($individual_fitness) as $subkey => $roulette) {
239             echo $roulette . ' ';
240             if ($random < $roulette) {
241                 echo ' [ ' . $subkey . ' ' . $roulette . ' ] ';
242                 $ret[$key] = $initial_populations[$subkey];
243                 break;
244             }
245         }
246         echo '<br>';
247     }
248     return $ret;
249 }
250 }
251
252 class CrossOver
253 {
254     public $population_size;
255     public $crossover_rate;
256
257     function __construct($population_size, $crossover_rate)
258     {
259         $this->population_size = $population_size;
260         $this->crossover_rate = $crossover_rate;
261     }
262
263     function selectedIndividual()
264     {
265         $randomZeroToOne = RandomNumbers::zeroToOne($this->population_size);
266         foreach ($randomZeroToOne as $key => $val) {
267             if ($val < $this->crossover_rate) {
268                 $ret[] = $key;
269             }
270         }
271         return $ret;
272     }
273 }
```

```
Explorer (Ctrl+Shift+E)
parcel.php > ...
269     }
270 }
271 return $ret;
272 }
273
274 function generateCombination($selectedIndividual)
275 {
276     foreach ($selectedIndividual as $val) {
277         $acak = $selectedIndividual[array_rand($selectedIndividual)];
278         $result[] = [$val, $acak];
279     }
280     return $result;
281 }
282
283 function combination($selectedIndividual)
284 {
285     $counter = 0;
286     while ($counter < 1) {
287         $combination = $this->generateCombination($selectedIndividual);
288         foreach ($combination as $val) {
289             $sum[] = count(array_unique($val));
290         }
291         if (array_sum($sum) === count($combination) * 2) {
292             return $combination;
293         }
294         $counter = 0;
295         $sum = [];
296     }
297 }
298
299 function cutPositions($parameters, $number_of_parents)
300 {
301     for ($i = 0; $i <= $number_of_parents - 1; $i++) {
302         $ret[] = Generate::randomGenLength($parameters);
303     }
304     return $ret;
305 }
306
307 function newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters)
308 {
309     $catalogue = Products::catalogue($parameters);
310     $offspring = [];
311
312     foreach ($combinations as $key => $combination) {
313         $individu1 = $selected_chromosome[$combination[0]];
314         $individu2 = $selected_chromosome[$combination[1]];
315
316         for ($i = 0; $i <= $catalogue['gen_length'] - 1; $i++) {
317             if ($i < $cut_positions[$key]) {
318                 $offspring[$key][] = $individu1[$i];
319             } else {
```

```
parcel.php X
parcel.php > --
318         $offspring[$key][] = $individual[$i];
319     } else {
320         $offspring[$key][] = $individual2[$i];
321     }
322 }
323 $ret[] = [
324     'index' => $selected_individuals[$key],
325     'offspring' => $offspring[$key]
326 ];
327 }
328 return $ret;
329 }
330
331 function updatePopulations($selected_chromosome, $new_individuals)
332 {
333     for ($i = 0; $i <= count($selected_chromosome) - 1; $i++) {
334         foreach ($new_individuals as $key => $individual) {
335             if ($i === $individual['index']) {
336                 $ret[$i][$key] = [
337                     'a' => 'new',
338                     'b' => $i
339                 ];
340             } else {
341                 $ret[$i][$key] = [
342                     'a' => 'old',
343                     'b' => $i
344                 ];
345             }
346         }
347     }
348
349     foreach ($ret as $key => $z) {
350         $index = array_search('new', array_column($z, 'a'));
351         if ($index || $index === 0) {
352             $index_new = array_search($z[$index]['b'], array_column($new_individuals, 'index'));
353             $result[] = $new_individuals[$index_new]['offspring'];
354         } else {
355             $result[] = $selected_chromosome[$key];
356         }
357     }
358     return $result;
359 }
360 }
361
362 $parameters = [
363     'file_name' => 'products.txt',
364     'indexes' => [0, 1],
365     'columns' => ['item', 'price'],
366     'population_size' => 10,
367     'fitness' => 1000,
```



```

parcel.php x
parcel.php > ...
363     'file_name' => 'products.txt',
364     'indexes' => [0, 1],
365     'columns' => ['item', 'price'],
366     'population_size' => 10,
367     'fitness' => 1000,
368     'max_generation' => 1000,
369     'crossover_rate' => 0.75,
370     'knapsack' => 150000
371 ];
372
373 $parameters = new Parameters($parameters);
374 $catalogue = Products::catalogue($parameters);
375
376 for ($i = 0; $i <= $parameters->max_generation; $i++) {
377     if ($i == 0) {
378         $initial_populations = Generate::initialPopulation($parameters);
379         $selected_product = Pairing::chromosomeGenWithProducts($initial_populations, $parameters);
380         echo '<p>';
381         $sumOfBuy = Calculate::sumOfBuy($selected_product);
382         print_r($sumOfBuy);
383         echo '<p>';
384         $individual_fitness = Fitness::evaluation($sumOfBuy, $parameters->knapsack);
385         print_r($individual_fitness);
386
387         echo 'Total fitness ' . array_sum($individual_fitness);
388         echo '<br>';
389
390         if (Fitness::isOne($individual_fitness)) {
391             echo 'Total fitness ' . array_sum($individual_fitness);
392             echo 'Maximum 1 achieved!';
393             $solution = Fitness::result($initial_populations, $individual_fitness, $sumOfBuy);
394             print_r($solution);
395             exit();
396         }
397         $solution = Fitness::result($initial_populations, $individual_fitness, $sumOfBuy);
398         print_r($solution);
399         $bests[] = $solution;
400
401         $select = new RouletteWheelSelection($parameters->population_size);
402         echo '<p>';
403         print_r($select->probability($individual_fitness));
404         echo '<p>';
405         $cumulative = $select->cummulative($individual_fitness);
406         echo '<p>';
407         print_r($cumulative);
408
409         $selected_chromosome = $select->selection($individual_fitness, $initial_populations);
410         echo '<p>';
411         echo 'Hasil seleksi<br>';
412         print_r($selected_chromosome);

```

```
parcel.php x
parcel.php > ...
411     echo 'Hasil seleksi<br>';
412     print_r($selected_chromosome);
413     echo '<br>';
414     $crossover = new Crossover($parameters->population_size, $parameters->crossover_rate);
415     $selected_individuals = $crossover->selectedIndividual($selected_chromosome, $initial_populations);
416     $number_of_parents = count($selected_individuals);
417     echo 'Individu terpilih: ';
418     echo $number_of_parents;
419     echo '&nbsp;';
420     print_r($selected_individuals);
421     echo '<p>Combination:<br>';
422     $combinations = $crossover->combination($selected_individuals);
423     print_r($combinations);
424     echo '<p>';
425     $cut_positions = $crossover->cutPositions($parameters, $number_of_parents);
426     echo 'Cut position:&nbsp;';
427     print_r($cut_positions);
428     echo '<br>';
429     $new_individuals = $crossover->newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters);
430     print_r($new_individuals);
431
432     echo '<p>';
433     echo 'New populations<br>';
434     $new_populations = $crossover->updatePopulations($selected_chromosome, $new_individuals);
435     print_r($new_populations);
436     echo '<p>';
437 }
438
439 if ($i > 0) {
440     echo 'Generasi: ' . $i . '<br>';
441     $selected_product = Pairing::chromosomeGenWithProducts($new_populations, $parameters);
442     echo '<p>';
443     $sumOfBuy = Calculate::sumOfBuy($selected_product);
444     print_r($sumOfBuy);
445     echo '<p>';
446     $individual_fitness = Fitness::evaluation($sumOfBuy, $parameters->knapsack);
447     print_r($individual_fitness);
448
449     echo ' Total fitness ' . array_sum($individual_fitness);
450     echo '<br>';
451
452     if (Fitness::isOne($individual_fitness)) {
453         echo ' Total fitness ' . array_sum($individual_fitness);
454         echo 'Maximum 1 achieved!';
455         $solution = Fitness::result($new_populations, $individual_fitness, $sumOfBuy);
456         print_r($solution);
457         exit();
458     }
459     $solution = Fitness::result($new_populations, $individual_fitness, $sumOfBuy);
460     print_r($solution);
461     $best_fit = $solution;
```

```
Explorer (Ctrl+Shift+E)
parcel.php > ...

464     echo '<p>';
465     print_r($select->probability($individual_fitness));
466     echo '<p>';
467     $cumulative = $select->cumulative($individual_fitness);
468     echo '<p>';
469     print_r($cumulative);
470
471     $selected_chromosome = $select->selection($individual_fitness, $initial_populations);
472     echo '<p>';
473     echo 'Hasil seleksi<br>';
474     print_r($selected_chromosome);
475     echo '<br>';
476     $crossover = new Crossover($parameters->population_size, $parameters->crossover_rate);
477     $selected_individuals = $crossover->selectedIndividual($selected_chromosome, $new_populations);
478     $number_of_parents = count($selected_individuals);
479     echo 'Individu terpilih: ';
480     echo $number_of_parents;
481     echo '&nbsp;';
482     print_r($selected_individuals);
483     echo '<p>Combination:<br>';
484     $combinations = $crossover->combination($selected_individuals);
485     print_r($combinations);
486     echo '<p>';
487     $cut_positions = $crossover->cutPositions($parameters, $number_of_parents);
488     echo 'Cut position:&nbsp;';
489     print_r($cut_positions);
490     echo '<br>';
491     $new_individuals = $crossover->newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters);
492     print_r($new_individuals);
493
494     echo '<p>';
495     echo 'New populations<br>';
496     $new_populations = $crossover->updatePopulations($selected_chromosome, $new_individuals);
497     print_r($new_populations);
498     echo '<p>';
499 }
500
501 echo '<p>';
502 echo '<p>Final: <br>';
503 print_r($bests);
504 echo '<br>';
505 $max_price = max(array_column($bests, 'price'));
506 $index = array_search($max_price, array_column($bests, 'price'));
507 echo '<p>';
508 print_r($bests[$index]);
509 $selected_product = Pairing::chromosomeGenWithProducts($bests[$index], $parameters);
510
```

```
localhost:teknikOptimasi/pertemuan7/parcel.php

Populations
0 101011101010000101000
1 010001110100111100100
2 010001100010111111101
3 010000110101101010100
4 010000110111000111100
5 1001111010001000000001
6 011000101111000111010
7 0101100011001000100001
8 00111111010010111001
9 111011101001100101011

0
Array ( [0] => Array ( [product] => Citrato 68 gr [price] => 8900 ) [1] => Array ( [product] => Botan Mackarel 425 gr [price] => 28900 ) [2] => Array ( [product] => Palm Fruit Kurma 500 gr [price] => 56900 ) [3] => Array ( [product] => Marjan Syrup 460 ml [price] => 17900 ) [4] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [5] => Array ( [product] => Kokola Wafer Cream 252 gr [price] => 18500 ) [6] => Array ( [product] => Pop Mie Pake Nasi 75 gr [price] => 8000 ) [7] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [8] => Array ( [product] => Kopi susu ABC [price] => 3000 ) ) 194390
1
Array ( [0] => Array ( [product] => Teh Soso Kotak [price] => 6900 ) [1] => Array ( [product] => Marjan Syrup 460 ml [price] => 17900 ) [2] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [3] => Array ( [product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500 ) [4] => Array ( [product] => Kokola Wafer Cream 252 gr [price] => 18500 ) [5] => Array ( [product] => Pop Mie Pake Nasi 75 gr [price] => 8000 ) [6] => Array ( [product] => Teh Hijau Kepala Djenggot 60 gr [price] => 11900 ) [7] => Array ( [product] => Madurasa 150 gr [price] => 16900 ) [8] => Array ( [product] => Makaroniku 200 gr [price] => 14900 ) [9] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [10] => Array ( [product] => Coca cola [price] => 15000 ) ) 184790
2
Array ( [0] => Array ( [product] => Teh Soso Kotak [price] => 6900 ) [1] => Array ( [product] => Marjan Syrup 460 ml [price] => 17900 ) [2] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [3] => Array ( [product] => Pop Mie Pake Nasi 75 gr [price] => 8000 ) [4] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [5] => Array ( [product] => Teh Hijau Kepala Djenggot 60 gr [price] => 11900 ) [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 Malkit Sali Cokelat 120 gr [price] => 4990 ) [10] => Array ( [product] => Kopi susu ABC [price] => 3000 ) [11] => Array ( [product] => Coca cola [price] => 15000 ) [12] => Array ( [product] => Danis biscuit [price] => 46900 ) ) 207330
3
Array ( [0] => Array ( [product] => Teh Soso Kotak [price] => 6900 ) [1] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [2] => Array ( [product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500 ) [3] => Array ( [product] => Sari Kacang Hijau 150 ml [price] => 7790 ) [4] => Array ( [product] => Teryaki Saen 275 ml [price] => 17900 ) [5] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [6] => Array ( [product] => Madurasa 150 gr [price] => 16900 ) [7] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [8] => Array ( [product] => Khong Guan Malkit Sali Cokelat 120 gr [price] => 4990 ) [9] => Array ( [product] => Coca cola [price] => 15000 ) ) 152920
4
Array ( [0] => Array ( [product] => Teh Soso Kotak [price] => 6900 ) [1] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [2] => Array ( [product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500 ) [3] => Array ( [product] => Sari Kacang Hijau 150 ml [price] => 7790 ) [4] => Array ( [product] => Pop Mie Pake Nasi 75 gr [price] => 8000 ) [5] => Array ( [product] => Teryaki Saen 275 ml [price] => 17900 ) [6] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [7] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [8] => Array ( [product] => Khong Guan Malkit Sali Cokelat 120 gr [price] => 4990 ) [9] => Array ( [product] => Coca cola [price] => 15000 ) ) 147020
5
Array ( [0] => Array ( [product] => Citrato 68 gr [price] => 8900 ) [1] => Array ( [product] => Keju Cheddar 180 gr [price] => 15250 ) [2] => Array ( [product] => Palm Fruit Kurma 500 gr [price] => 56900 ) [3] => Array ( [product] => Marjan Syrup 460 ml [price] => 17900 ) [4] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [5] => Array ( [product] => Kokola Wafer Cream 252 gr [price] => 18500 ) [6] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [7] => Array ( [product] => Danis biscuit [price] => 46900 ) ) 205390
6
Array ( [0] => Array ( [product] => Botan Mackarel 425 gr [price] => 28900 ) [1] => Array ( [product] => Botan Mackarel 425 gr [price] => 28900 ) [2] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [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] => Teryaki Saen 275 ml [price] => 17900 ) [7] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [8] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [9] => Array ( [product] => Khong Guan Malkit Sali Cokelat 120 gr [price] => 4990 ) [10] => Array ( [product] => Kopi susu ABC [price] => 3000 ) [11] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [12] => Array ( [product] => Khong Guan Malkit Sali Cokelat 120 gr [price] => 4990 ) [13] => Array ( [product] => Kopi susu ABC [price] => 3000 ) [14] => Array ( [product] => Danis biscuit [price] => 46900 ) ) 315370
7
Array ( [0] => Array ( [product] => Teh Soso Kotak [price] => 6900 ) [1] => Array ( [product] => Keju Cheddar 180 gr [price] => 15250 ) [2] => Array ( [product] => Palm Fruit Kurma 500 gr [price] => 56900 ) [3] => Array ( [product] => Kokola Wafer Cream 252 gr [price] => 18500 ) [4] => Array ( [product] => Sari Kacang Hijau 150 ml [price] => 7790 ) [5] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [6] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [7] => Array ( [product] => Danis biscuit [price] => 46900 ) ) 180790
8
Array ( [0] => Array ( [product] => Botan Mackarel 425 gr [price] => 28900 ) [1] => Array ( [product] => Keju Cheddar 180 gr [price] => 15250 ) [2] => Array ( [product] => Palm Fruit Kurma 500 gr [price] => 56900 ) [3] => Array ( [product] => Marjan Syrup 460 ml [price] => 17900 ) [4] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [5] => Array ( [product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500 ) [6] => Array ( [product] => Kokola Wafer Cream 252 gr [price] => 18500 ) [7] => Array ( [product] => Sari Kacang Hijau 150 ml [price] => 7790 ) [8] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [9] => Array ( [product] => Madurasa 150 gr [price] => 16900 ) [10] => Array ( [product] => Makaroniku 200 gr [price] => 14900 ) [11] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [12] => Array ( [product] => Khong Guan Malkit Sali Cokelat 120 gr [price] => 4990 ) [13] => Array ( [product] => Kopi susu ABC [price] => 3000 ) [14] => Array ( [product] => Danis biscuit [price] => 46900 ) ) 315370
9
Array ( [0] => Array ( [product] => Citrato 68 gr [price] => 8900 ) [1] => Array ( [product] => Teh Soso Kotak [price] => 6900 ) [2] => Array ( [product] => Botan Mackarel 425 gr [price] => 28900 ) [3] => Array ( [product] => Palm Fruit Kurma 500 gr [price] => 56900 ) [4] => Array ( [product] => 365 Wafer Stick 500 gr [price] => 32390 ) [5] => Array ( [product] => Nissin Biscuit Lemonia Twist 340 gr [price] => 22500 ) [6] => Array ( [product] => Kokola Wafer Cream 252 gr [price] => 18500 ) [7] => Array ( [product] => Sari Kacang Hijau 150 ml [price] => 7790 ) [8] => Array ( [product] => Dua Belas Sambal 135:340:535 gr [price] => 8650 ) [9] => Array ( [product] => Teh Hijau Kepala Djenggot 60 gr [price] => 11900 ) [10] => Array ( [product] => Ultra Low Fat Susu UHT 1000 ml [price] => 19900 ) [11] => Array ( [product] => Kopi susu ABC
```

