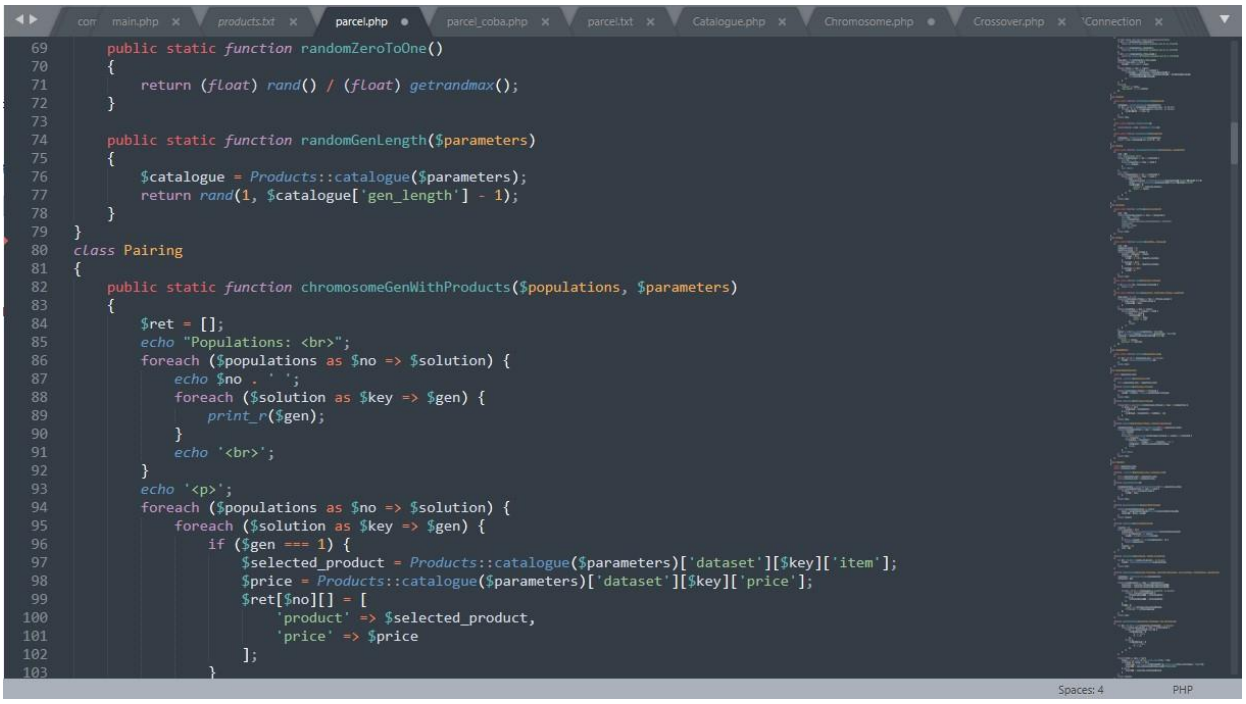
NAMA : IRMA ERYANTI PUTRI

NIM : 1800018272

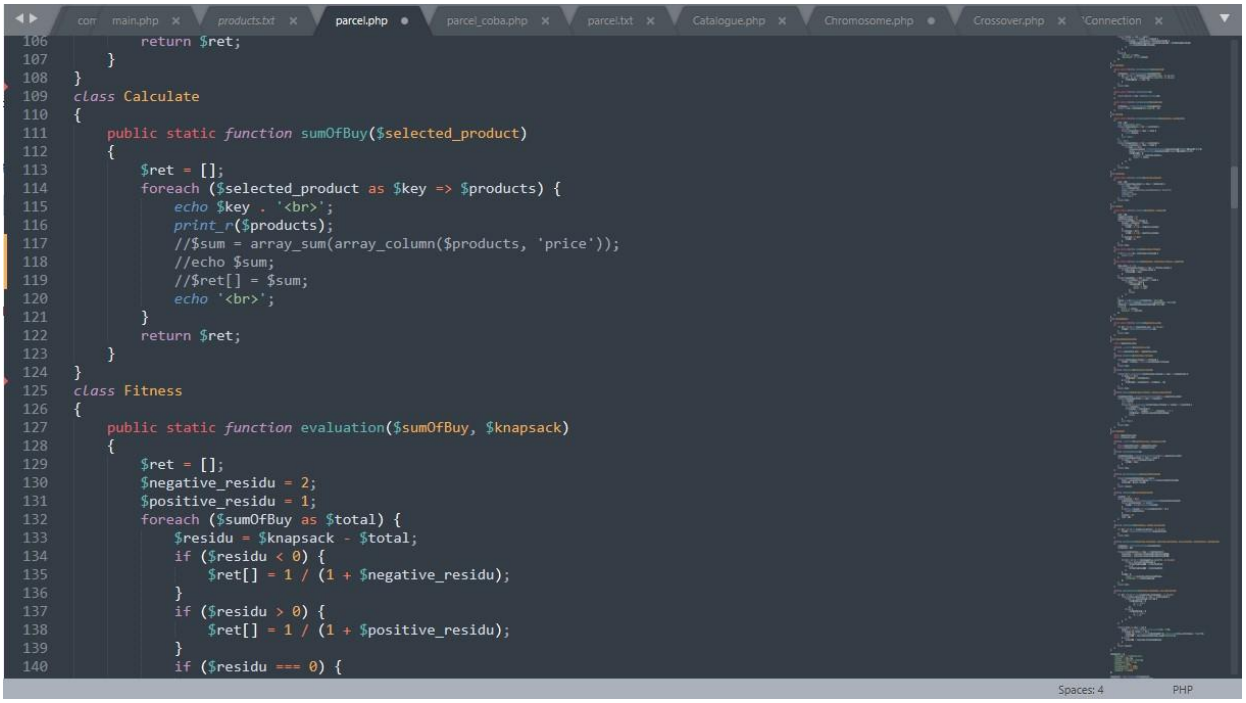
INHAL 10 PRAKTIKUM TEKNIK OPTIMASI

File parcel.php

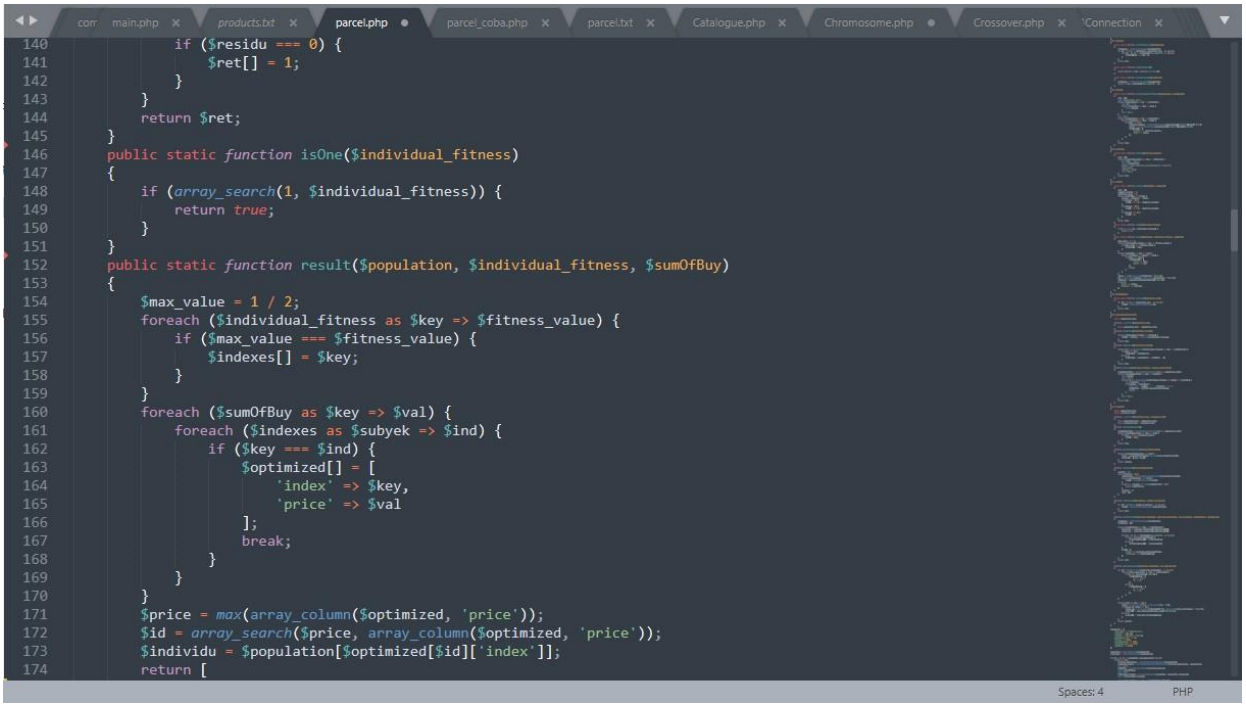
```php
<?php
class Parameters
{
    public $file_name;
    public $indexes;
    public $columns;
    public $population_size;
    public $fitness;
    public $max_generation;
    public $knapsack;
    public $crossover_rate;
    public function __construct($parameters)
    {
        $this->file_name = $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($parameters->file_name)) {
        }
        if (!is_string($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;
    }

```



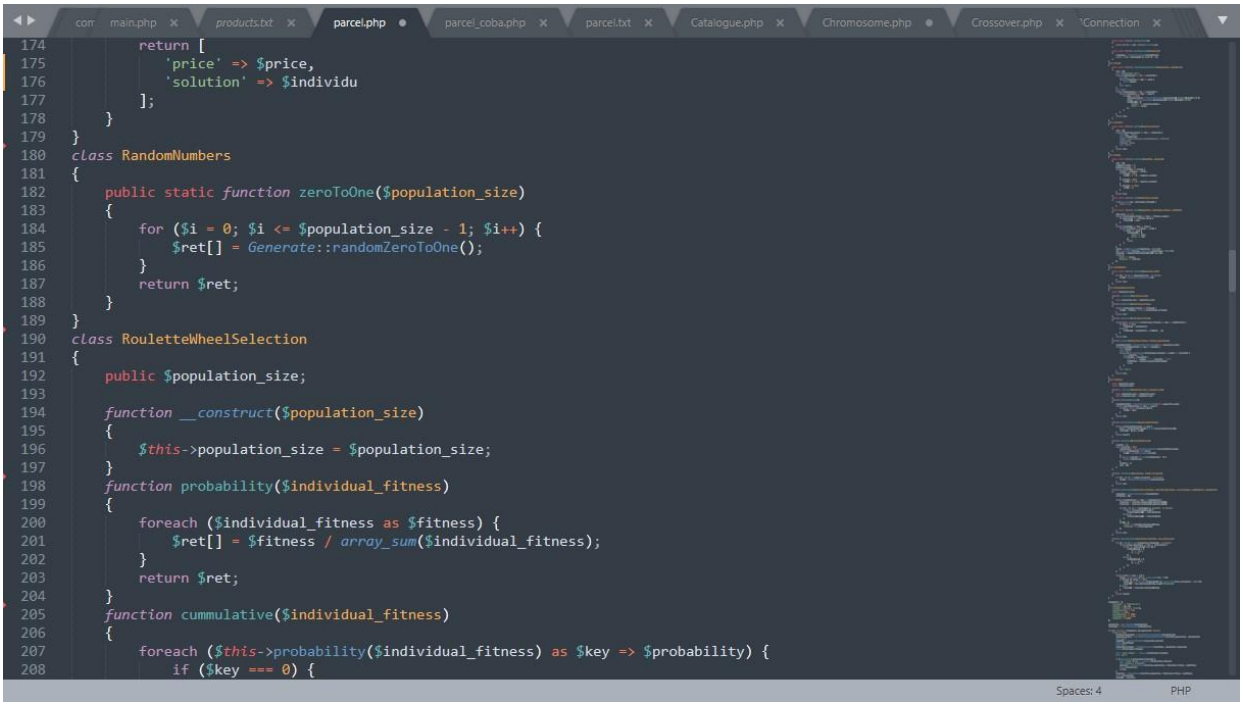

```php
    public static function randomZeroToOne()
    {
        return (float) rand() / (float) getrandmax();
    }

    public static function randomGenLength($parameters)
    {
        $catalogue = Products::catalogue($parameters);
        return rand(1, $catalogue['gen_length'] - 1);
    }
}
class Pairing
{
    public static function chromosomeGenWithProducts($populations, $parameters)
    {
        $ret = [];
        echo "Populations: <br>";
        foreach ($populations as $no => $solution) {
            echo $no . ' ';
            foreach ($solution as $key => $gen) {
                print_r($gen);
            }
            echo '<br>';
        }
        echo '<p>';
        foreach ($populations as $no => $solution) {
            foreach ($solution as $key => $gen) {
                if ($gen === 1) {
                    $selected_product = Products::catalogue($parameters)['dataset'][$key]['item'];
                    $price = Products::catalogue($parameters)['dataset'][$key]['price'];
                    $ret[$no][] = [
                        'product' => $selected_product,
                        'price' => $price
                    ];
                }
```



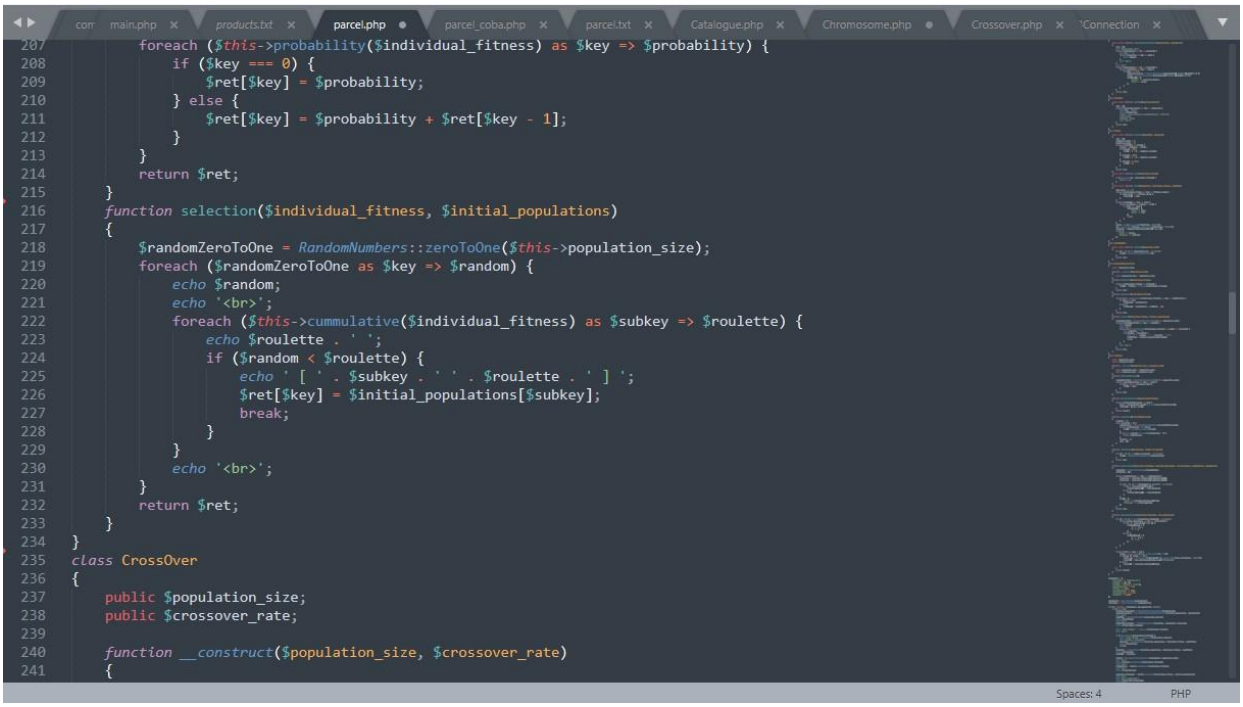

```php
        return $ret;
    }
}
class Calculate
{
    public static function sumOfBuy($selected_product)
    {
        $ret = [];
        foreach ($selected_product as $key => $products) {
            echo $key . '<br>';
            print_r($products);
            //$sum = array_sum(array_column($products, 'price'));
            //echo $sum;
            //$ret[] = $sum;
            echo '<br>';
        }
        return $ret;
    }
}
class Fitness
{
    public static function evaluation($sumOfBuy, $knapsack)
    {
        $ret = [];
        $negative_residu = 2;
        $positive_residu = 1;
        foreach ($sumOfBuy as $total) {
            $residu = $knapsack - $total;
            if ($residu < 0) {
                $ret[] = 1 / (1 + $negative_residu);
            }
            if ($residu > 0) {
                $ret[] = 1 / (1 + $positive_residu);
            }
            if ($residu === 0) {
```



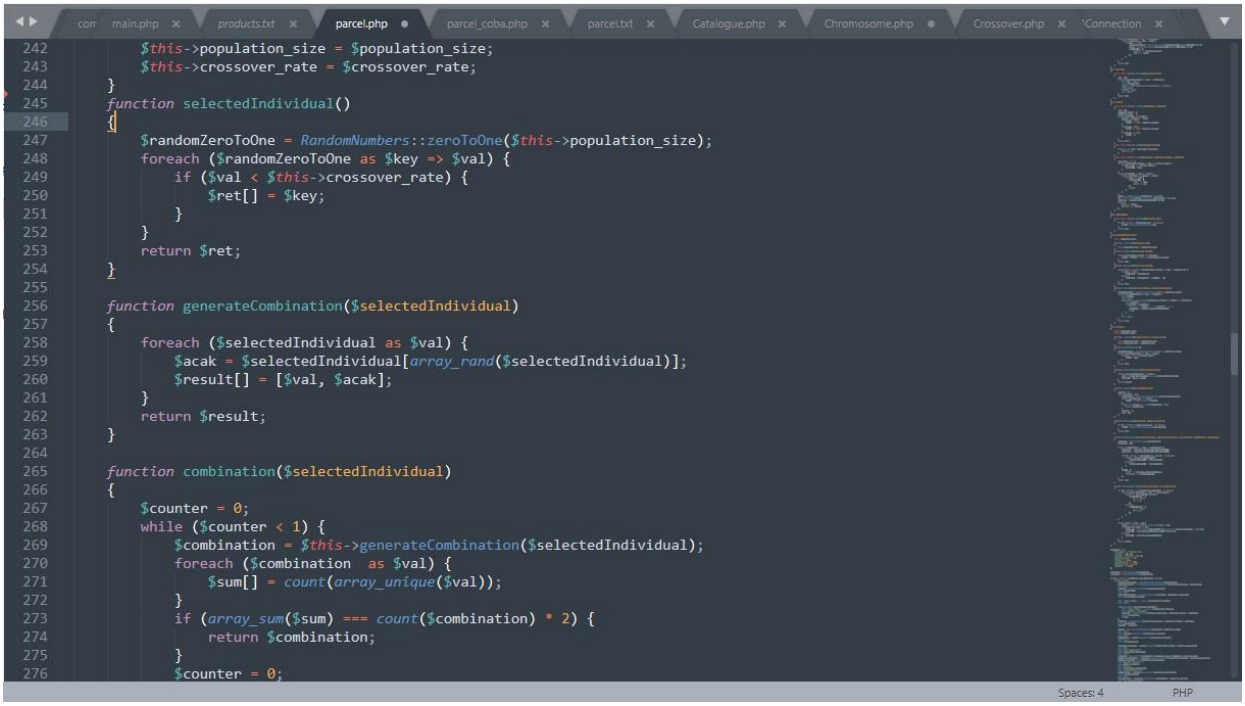

```php
            if ($residu === 0) {
                $ret[] = 1;
            }
        }
        return $ret;
    }
    public static function isOne($individual_fitness)
    {
        if (array_search(1, $individual_fitness)) {
            return true;
        }
    }
    public static function result($population, $individual_fitness, $sumOfBuy)
    {
        $max_value = 1 / 2;
        foreach ($individual_fitness as $key => $fitness_value) {
            if ($max_value === $fitness_value) {
                $indexes[] = $key;
            }
        }
        foreach ($sumOfBuy as $key => $val) {
            foreach ($indexes as $subyek => $ind) {
                if ($key === $ind) {
                    $optimized[] = [
                        'index' => $key,
                        'price' => $val
                    ];
                    break;
                }
            }
        }
        $price = max(array_column($optimized, 'price'));
        $id = array_search($price, array_column($optimized, 'price'));
        $individu = $population[$optimized[$id]['index']];
        return [
```



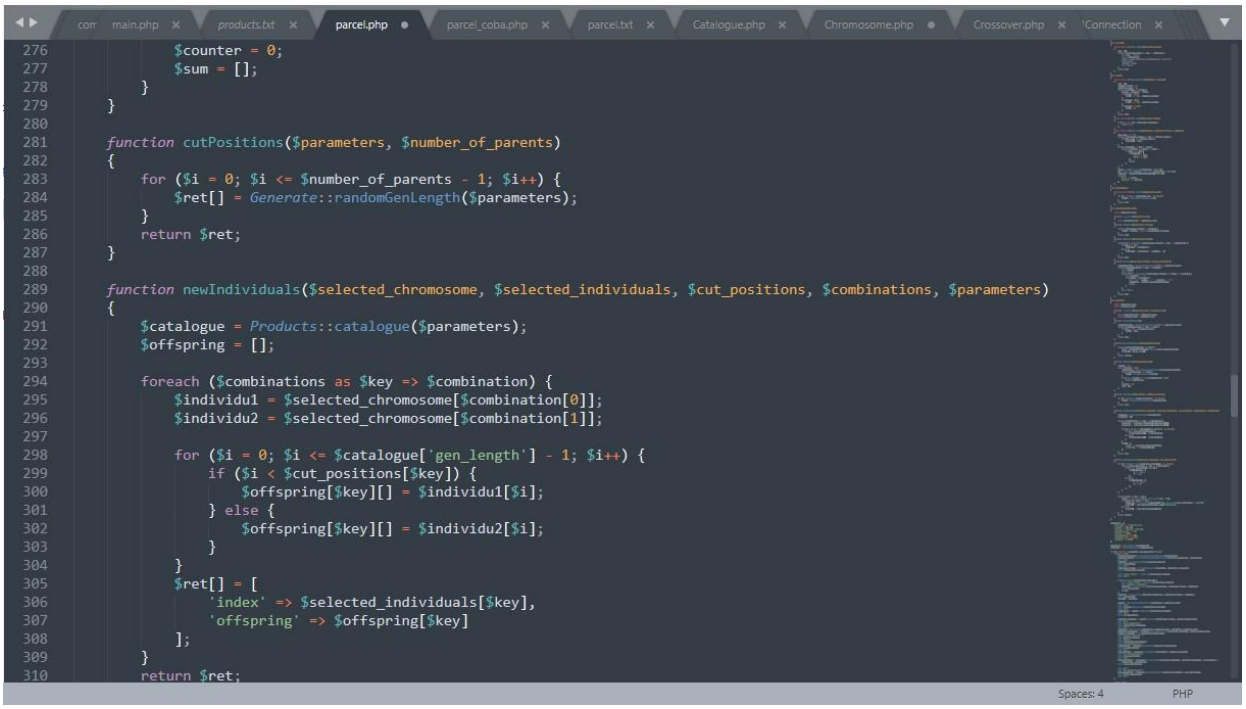

```php
        return [
            'price' => $price,
            'solution' => $individu
        ];
    }
}
class RandomNumbers
{
    public static function zeroToOne($population_size)
    {
        for ($i = 0; $i <= $population_size - 1; $i++) {
            $ret[] = Generate::randomZeroToOne();
        }
        return $ret;
    }
}
class RouletteWheelSelection
{
    public $population_size;

    function __construct($population_size)
    {
        $this->population_size = $population_size;
    }
    function probability($individual_fitness)
    {
        foreach ($individual_fitness as $fitness) {
            $ret[] = $fitness / array_sum($individual_fitness);
        }
        return $ret;
    }
    function cummulative($individual_fitness)
    {
        foreach ($this->probability($individual_fitness) as $key => $probability) {
            if ($key === 0) {
```



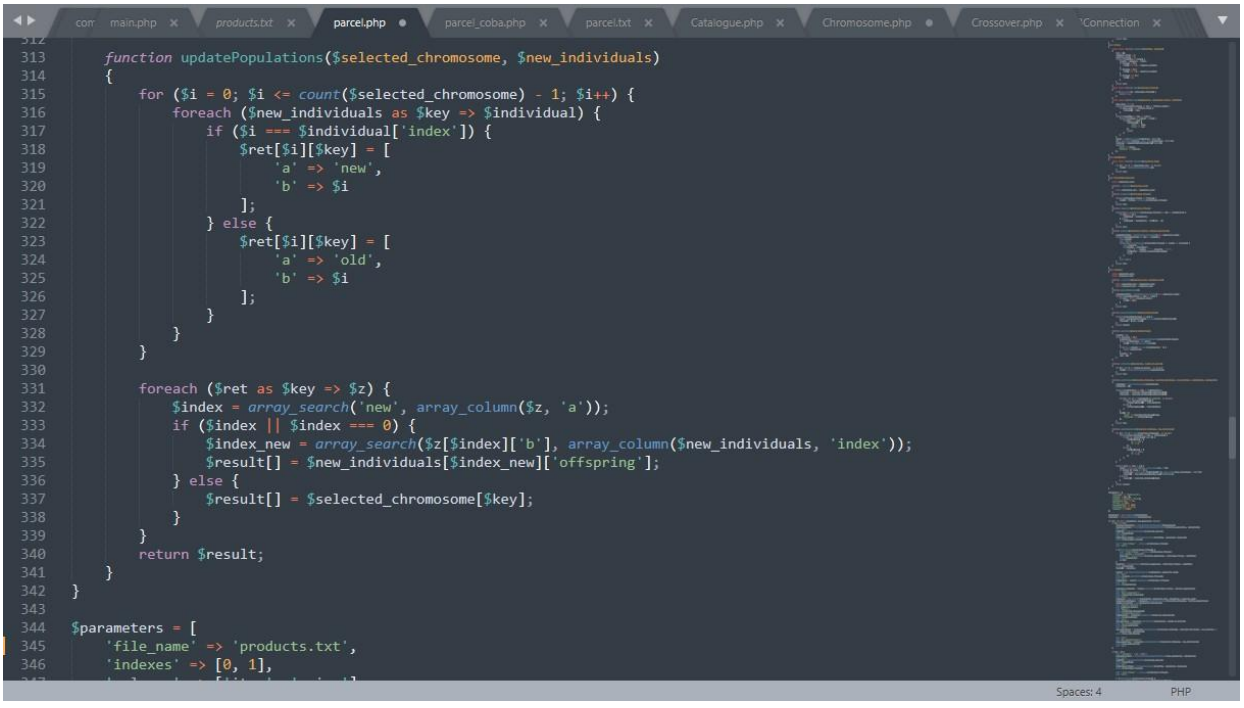

```php
        foreach ($this->probability($individual_fitness) as $key => $probability) {
            if ($key === 0) {
                $ret[$key] = $probability;
            } else {
                $ret[$key] = $probability + $ret[$key - 1];
            }
        }
        return $ret;
    }
    function selection($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
            $counter = 0;
            $sum = [];
        }
    }

    function cutPositions($parameters, $number_of_parents)
    {
        for ($i = 0; $i <= $number_of_parents - 1; $i++) {
            $ret[] = Generate::randomGenLength($parameters);
        }
        return $ret;
    }

    function newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $combinations, $parameters)
    {
        $catalogue = Products::catalogue($parameters);
        $offspring = [];

        foreach ($combinations as $key => $combination) {
            $individu1 = $selected_chromosome[$combination[0]];
            $individu2 = $selected_chromosome[$combination[1]];

            for ($i = 0; $i <= $catalogue['gen_length'] - 1; $i++) {
                if ($i < $cut_positions[$key]) {
                    $offspring[$key][] = $individu1[$i];
                } else {
                    $offspring[$key][] = $individu2[$i];
                }
            }
            $ret[] = [
                'index' => $selected_individuals[$key],
                'offspring' => $offspring[$key]
            ];
        }
        return $ret;
```




```php
    function updatePopulations($selected_chromosome, $new_individuals)
    {
        for ($i = 0; $i <= count($selected_chromosome) - 1; $i++) {
            foreach ($new_individuals as $key => $individual) {
                if ($i === $individual['index']) {
                    $ret[$i][$key] = [
                        'a' => 'new',
                        'b' => $i
                    ];
                } else {
                    $ret[$i][$key] = [
                        'a' => 'old',
                        'b' => $i
                    ];
                }
            }
        }

        foreach ($ret as $key => $z) {
            $index = array_search('new', array_column($z, 'a'));
            if ($index || $index === 0) {
                $index_new = array_search($z[$index]['b'], array_column($new_individuals, 'index'));
                $result[] = $new_individuals[$index_new]['offspring'];
            } else {
                $result[] = $selected_chromosome[$key];
            }
        }
        return $result;
    }
}

$parameters = [
    'file_name' => 'products.txt',
    'indexes' => [0, 1],
```

```php
$parameters = [
    'file_name' => 'products.txt',
    'indexes' => [0, 1],
    'columns' => ['item', 'price'],
    'population_size' => 10,
    'fitness' => 1000,
    'max_generation' => 1000,
    'crossover_rate' => 0.75,
    'knapsack' => 150000
];

$parameters = new Parameters($parameters);
$catalogue = Products::catalogue($parameters);

for ($i = 0; $i <= $parameters->max_generation; $i++) {
    if ($i === 0) {
        $initial_populations = Generate::initialPopulation($parameters);
        $selected_product = Pairing::chromosomeGenWithProducts($initial_populations, $parameters);
        echo '<p>';
        $sumOfBuy = Calculate::sumOfBuy($selected_product);
        print_r($sumOfBuy);
        echo '<p>';
        $individual_fitness = Fitness::evaluation($sumOfBuy, $parameters->knapsack);
        print_r($individual_fitness);

        echo ' Total fitness ' . array_sum($individual_fitness);
        echo '<br>';

        if (Fitness::isOne($individual_fitness)) {
            echo ' Total fitness ' . array_sum($individual_fitness);
            echo 'Maximum 1 achieved!';
            $solution = Fitness::result($initial_populations, $individual_fitness, $sumOfBuy);
            print_r($solution);
            exit();
        }
```

```php
        $solution = Fitness::result($initial_populations, $individual_fitness, $sumOfBuy);
        print_r($solution);
        $bests[] = $solution;

        $select = new RouletteWheelSelection($parameters->population_size);
        echo '<p>';
        print_r($select->probability($individual_fitness));
        echo '<p>';
        $cummulative = $select->cummulative($individual_fitness);
        echo '<p>';
        print_r($cummulative);

        $selected_chromosome = $select->selection($individual_fitness, $initial_populations);
        echo '<p>';
        echo 'Hasil seleksi<br>';
        print_r($selected_chromosome);
        echo '<br>';
        $crossover = new CrossOver($parameters->population_size, $parameters->crossover_rate);
        $selected_individuals = $crossover->selectedIndividual($selected_chromosome, $initial_populations);
        $number_of_parents = count($selected_individuals);
        echo 'Individu terpilih: ';
        echo $number_of_parents;
        echo ' ';
        print_r($selected_individuals);
        echo '<p>Combination:<br>';
        $combinations = $crossover->combination($selected_individuals);
        print_r($combinations);
        echo '<p>';
        $cut_positions = $crossover->cutPositions($parameters, $number_of_parents);
        echo 'Cut position: ';
        print_r($cut_positions);
        echo '<br>';
        $new_individuals = $crossover->newIndividuals($selected_chromosome, $selected_individuals, $cut_positions, $
            combinations, $parameters);
```

```
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);
```

Spaces: 4 PHP

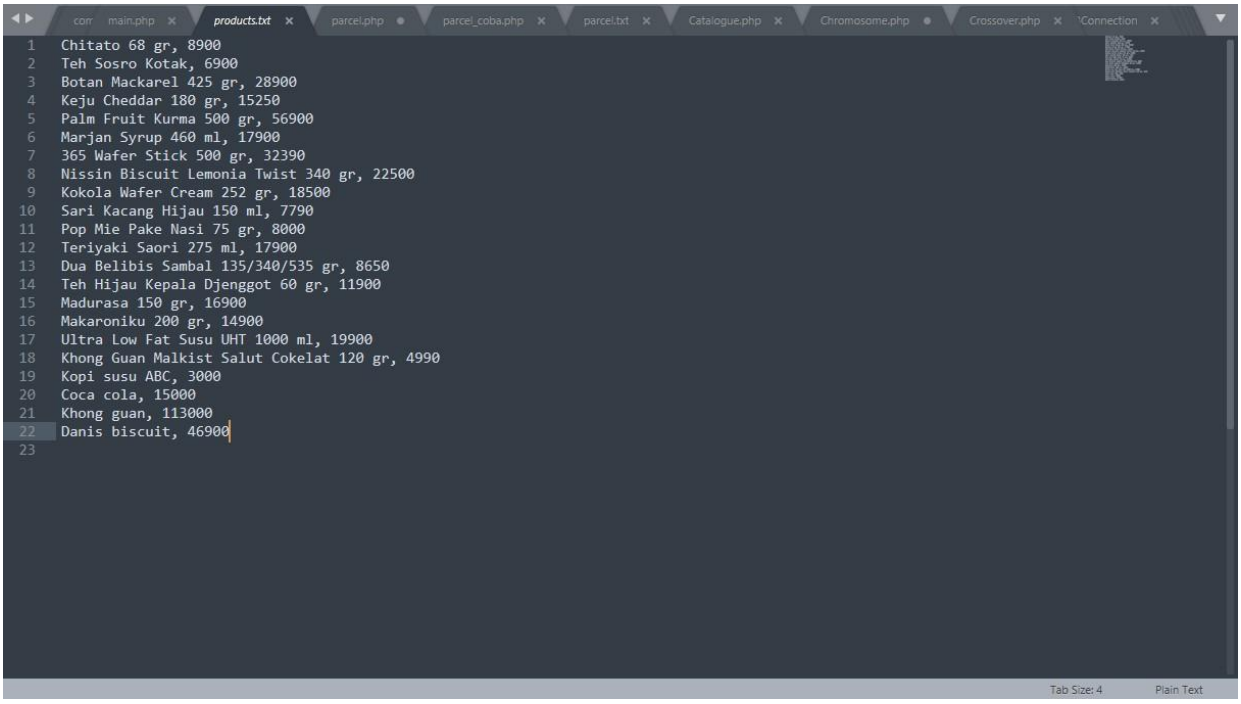
```
444
445         $select = new RouletteWheelSelection($parameters->population_size);
446         echo '<p>';
447         print_r($select->probability($individual_fitness));
448         echo '<p>';
449         $cummulative = $select->cummulative($individual_fitness);
450         echo '<p>';
451         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);
```

Spaces: 4 PHP

File products.txt



Output:

Populations:
0 10100101110100010100 11
1 11010011010000001100 10
2 00011011000000101101 00
3 11010010001100000010 11
4 00101000000100010101 01
5 10000011001001111000 01
6 10110100101000001001 01
7 01110100000110010000 01
8 10001000111010111000 11
9 10010100111001100011 1

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.