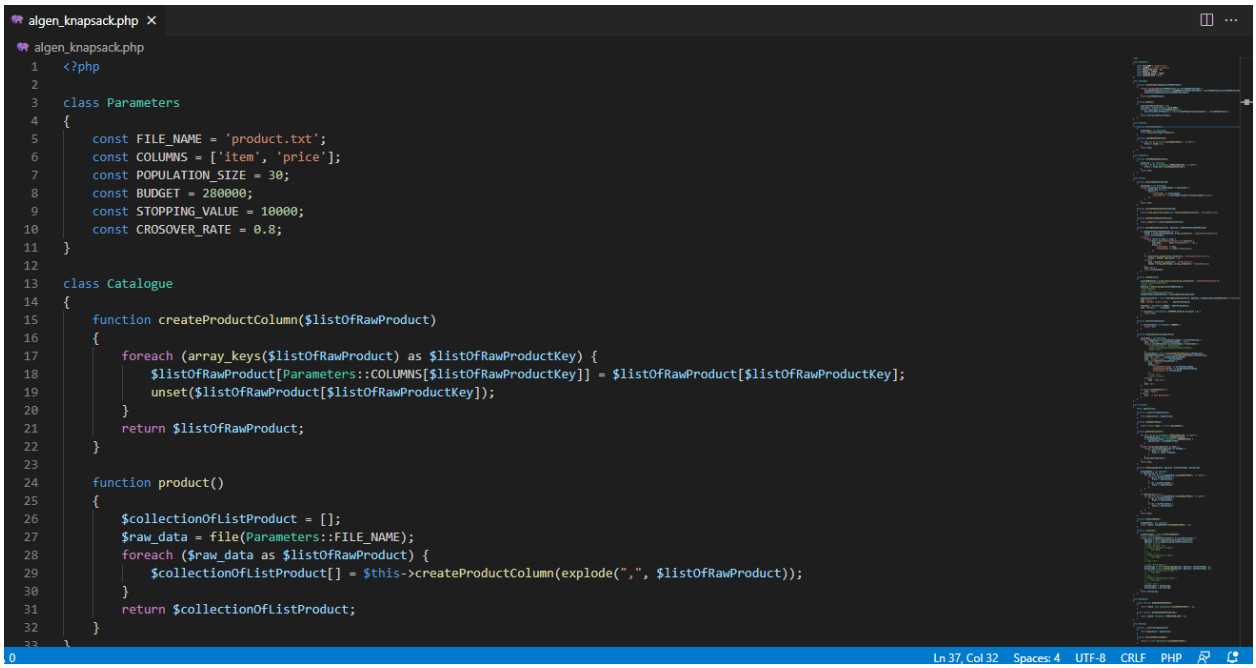
NAMA : IRMA ERYANTI PUTRI

NIM : 1800018272

PRAKTIKUM 9 TEKNIK OPTIMASI

Algen_knapsack.php

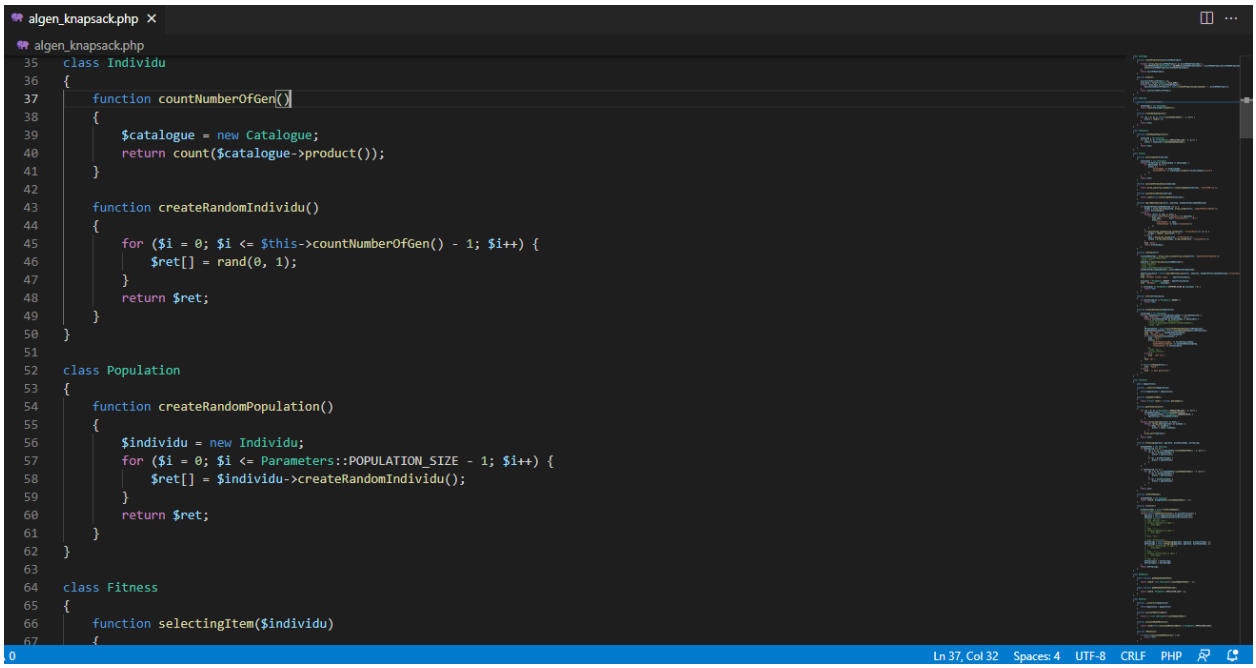


```php
<?php

class Parameters
{
    const FILE_NAME = 'product.txt';
    const COLUMNS = ['item', 'price'];
    const POPULATION_SIZE = 30;
    const BUDGET = 280000;
    const STOPPING_VALUE = 10000;
    const CROSOVER_RATE = 0.8;
}

class Catalogue
{
    function createProductColumn($listOfRawProduct)
    {
        foreach (array_keys($listOfRawProduct) as $listOfRawProductKey) {
            $listOfRawProduct[Parameters::COLUMNS[$listOfRawProductKey]] = $listOfRawProduct[$listOfRawProductKey];
            unset($listOfRawProduct[$listOfRawProductKey]);
        }
        return $listOfRawProduct;
    }

    function product()
    {
        $collectionOfListProduct = [];
        $raw_data = file(Parameters::FILE_NAME);
        foreach ($raw_data as $listOfRawProduct) {
            $collectionOfListProduct[] = $this->createProductColumn(explode(",", $listOfRawProduct));
        }
        return $collectionOfListProduct;
    }
}
```



```php
class Individu
{
    function countNumberOfGen()
    {
        $catalogue = new Catalogue;
        return count($catalogue->product());
    }

    function createRandomIndividu()
    {
        for ($i = 0; $i <= $this->countNumberOfGen() - 1; $i++) {
            $ret[] = rand(0, 1);
        }
        return $ret;
    }
}

class Population
{
    function createRandomPopulation()
    {
        $individu = new Individu;
        for ($i = 0; $i <= Parameters::POPULATION_SIZE - 1; $i++) {
            $ret[] = $individu->createRandomIndividu();
        }
        return $ret;
    }
}

class Fitness
{
    function selectingItem($individu)
    {
```

algen_knapsack.php

```php
class Fitness
{
    function selectingItem($individu)
    {
        $catalogue = new Catalogue;
        foreach ($individu as $individuKey => $binaryGen) {
            if ($binaryGen === 1) {
                $ret[] = [
                    'selectedKey' => $individuKey,
                    'selectedPrice' => $catalogue->product()[$individuKey]['price']
                ];
            }
        }
        return $ret;
    }

    function calculateFitnessValue($individu)
    {
        return array_sum(array_column($this->selectingItem($individu), 'selectedPrice'));
    }

    function countSelectedItem($individu)
    {
        return count($this->selectingItem($individu));
    }

    function searchBestIndividu($fits, $maxItem, $numberOfIndividuHasMaxItem)
    {
        if ($numberOfIndividuHasMaxItem === 1) {
            $index = array_search($maxItem, array_column($fits, 'numberOfSelectedItem'));
            return $fits[$index];
        } else {
```

0 Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

algen_knapsack.php

```php
            return $fits[$index];
        } else {
            foreach ($fits as $key => $val) {
                if ($val['numberOfSelectedItem'] === $maxItem) {
                    echo $key . ' ' . $val['fitnessValue'] . '<br>';
                    $ret[] = [
                        'individuKey' => $key,
                        'fitnessValue' => $val['fitnessValue']
                    ];
                }
            }
            if (count(array_unique(array_column($ret, 'fitnessValue'))) === 1) {
                $index = rand(0, count($ret) - 1);
            } else {
                $max = max(array_column($ret, 'fitnessValue'));
                $index = array_search($max, array_column($ret, 'fitnessValue'));
            }
            echo 'Hasil';
            return $ret[$index];
        }
    }

    function isFound($fits)
    {
        $countedMaxItems = array_count_values(array_column($fits, 'numberOfSelectedItem'));
        //print_r($countedMaxItems);
        //echo '<br>';
        $maxItem = max(array_keys($countedMaxItems));
        //echo $maxItem;
        //echo '<br>';
        //echo $countedMaxItems[$maxItem];
        $numberOfIndividuHasMaxItem = $countedMaxItems[$maxItem];
```

0 Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

algen_knapsack.php

```php
        $bestFitnessValue = $this->searchBestIndividu($fits, $maxItem, $numberOfIndividuHasMaxItem)['fitnessValue'];
        echo '<br>';
        echo '<br>Best fitness value: ' . $bestFitnessValue;

        $residual = Parameters::BUDGET - $bestFitnessValue;
        echo ' Residual: ' . $residual;

        if ($residual <= Parameters::STOPPING_VALUE && $residual > 0) {
            return TRUE;
        }
    }

    function isFit($fitnessValue)
    {
        if ($fitnessValue <= Parameters::BUDGET) {
            return TRUE;
        }
    }

    function fitnessEvaluation($population)
    {
        $catalogue = new Catalogue;
        foreach ($population as $listOfIndividuKey => $listOfIndividu) {
            echo 'Individu-' . $listOfIndividuKey . '<br>';
            foreach ($listOfIndividu as $individuKey => $binaryGen) {
                //echo $binaryGen . '  ';
                //print_r($catalogue->product()[$individuKey]);
                //echo '<br>';
            }
            $fitnessValue = $this->calculateFitnessValue($listOfIndividu);
            $numberOfSelectedItem = $this->countSelectedItem($listOfIndividu);
```

algen_knapsack.php

```php
            $fitnessValue = $this->calculateFitnessValue($listOfIndividu);
            $numberOfSelectedItem = $this->countSelectedItem($listOfIndividu);
            echo 'Max. Item: ' . $numberOfSelectedItem;
            echo ' Fitness value: ' . $fitnessValue;
            if ($this->isFit($fitnessValue)) {
                echo ' (Fit)';
                $fits[] = [
                    'selectedIndividuKey' => $listOfIndividuKey,
                    'numberOfSelectedItem' => $numberOfSelectedItem,
                    'fitnessValue' => $fitnessValue
                ];
                //echo '<p>';
                //print_r($fits);
            } else {
                echo ' (Not Fit)';
            }
            echo '<p>';
        }

        if ($this->isFound($fits)) {
            echo ' Found';
        } else {
            echo ' >> Next generation';
        }
    }
}

class Crossover
{
    public $populations;

    function __construct($populations)
```

algen_knapsack.php

```php
        if ($offspring === 1) {
            for ($i = 0; $i <= $lengthOfGen->countNumberOfGen() - 1; $i++) {
                if ($i <= $cutPointIndex) {
                    $ret[] = $parent1[$i];
                }
                if ($i > $cutPointIndex) {
                    $ret[] = $parent2[$i];
                }
            }
        }

        if ($offspring === 2) {
            for ($i = 0; $i <= $lengthOfGen->countNumberOfGen() - 1; $i++) {
                if ($i <= $cutPointIndex) {
                    $ret[] = $parent2[$i];
                }
                if ($i > $cutPointIndex) {
                    $ret[] = $parent1[$i];
                }
            }
        }
        return $ret;
    }

    function cutPointRandom()
    {
        $lengthOfGen = new Individu;
        return rand(0, $lengthOfGen->countNumberOfGen() - 1);
    }

    function crossover()
    {
        $cutPointIndex = $this->cutPointRandom();
```

0 Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

algen_knapsack.php

```php
        $cutPointIndex = $this->cutPointRandom();
        //echo $cutPointIndex;
        foreach ($this->generateCrossover() as $listOfCrossover) {
            $parent1 = $this->populations[$listOfCrossover[0]];
            $parent2 = $this->populations[$listOfCrossover[1]];
            // echo '<p></p>';
            // echo 'Parents :<br>';
            // foreach ($parent1 as $gen) {
            //     echo $gen;
            // }
            // echo ' >< ';
            // foreach ($parent2 as $gen) {
            //     echo $gen;
            // }
            // echo '<br>';

            // echo 'Offspring<br>';
            $offspring1 = $this->offspring($parent1, $parent2, $cutPointIndex, 1);
            $offspring2 = $this->offspring($parent1, $parent2, $cutPointIndex, 2);
            // foreach ($offspring1 as $gen) {
            //     echo $gen;
            // }
            // echo ' >< ';
            // foreach ($offspring2 as $gen) {
            //     echo $gen;
            // }
            // echo '<br>';
            $offsprings[] = $offspring1;
            $offsprings[] = $offspring2;
        }
        return $offsprings;
    }
}
```

0 Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

algen_knapsack.php

```php
class Randomizer
{
    static function getRandomIndexOfGen()
    {
        return rand(0, (new Individu())->countNumberOfGen() - 1);
    }

    static function getRandomIndexOfIndividu()
    {
        return rand(0, Parameters::POPULATION_SIZE - 1);
    }
}

class Mutation
{
    function __construct($population)
    {
        $this->population = $population;
    }

    function calculateMutationRate()
    {
        return 1 / (new Individu())->countNumberOfGen();
    }

    function calculateNumOfMutation()
    {
        return round($this->calculateMutationRate() * Parameters::POPULATION_SIZE);
    }

    function isMutation()
    {
        if ($this->calculateNumOfMutation() > 0){
            return TRUE;
        }
    }

    function generateMutation($valueOfGen)
    {
        if ($valueOfGen === 0){
            return 1;
        } else {
            return 0;
        }
    }

    function mutation()
    {
        if ($this->isMutation()){
            for ($i = 0; $i <= $this->calculateNumOfMutation() - 1; $i++) {
                $indexOfIndividu = Randomizer::getRandomIndexOfIndividu();
                $indexOfGen = Randomizer::getRandomIndexOfGen();
                $selectedIndividu = $this->population[$indexOfIndividu];

                //echo 'Before mutation: ';
                //print_r($selectedIndividu);
                //echo '<br>';
                $valueOfGen = $selectedIndividu[$indexOfGen];
                $mutatedGen = $this->generateMutation($valueOfGen);
                $selectedIndividu[$indexOfGen] = $mutatedGen;
                //echo 'After mutation: ';
                //print_r($selectedIndividu);
                $ret[] = $selectedIndividu;
            }
```

algen_knapsack.php

```
                $valueOfGen = $selectedIndividu[$indexOfGen];
                $mutatedGen = $this->generateMutation($valueOfGen);
                $selectedIndividu[$indexOfGen] = $mutatedGen;
                //echo 'After mutation: ';
                //print_r($selectedIndividu);
                $ret[] = $selectedIndividu;
            }
            return $ret;
        }
    }
}

class Selection
{
    function __construct($population, $combinedOffsprings)
    {
        $this->population = $population;
        $this->combinedOffsprings = $combinedOffsprings;
    }

    function createTemporaryPopulation()
    {
        foreach ($this->combinedOffsprings as $offspring){
            $this->population[] = $offspring;
        }
        return $this->population;
    }

    function getVariableValue($basePopulation, $fitTemporaryPopulation)
    {
        foreach ($fitTemporaryPopulation as $val){
            $ret[] = $basePopulation[$val[1]];
```

0 Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

algen_knapsack.php

```
    function sortFitTemporaryPopulation()
    {
        $tempPopulation = $this->createTemporaryPopulation();
        $fitness = new Fitness;
        foreach ($tempPopulation as $key => $individu){
            $fitnessValue = $fitness->calculateFitnessValue($individu);
            if ($fitness->isFit($fitnessValue)){
                $fitTemporaryPopulation[] = [
                    $fitnessValue,
                    $key
                ];
            }
        }
        rsort($fitTemporaryPopulation);
        $fitTemporaryPopulation = array_slice($fitTemporaryPopulation, 0, Parameters::POPULATION_SIZE);
        return $this->getVariableValue($tempPopulation, $fitTemporaryPopulation);
    }

    function selectingIndividus()
    {
        $selected = $this->sortFitTemporaryPopulation();
        echo '<p></p>';
        print_r($selected);
    }

}

$initalPopulation = new Population;
$population = $initalPopulation->createRandomPopulation();

$fitness = new Fitness;
$fitness->fitnessEvaluation($population);
```

0 Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

algen_knapsack.php

```php

$fitness = new Fitness;
$fitness->fitnessEvaluation($population);

$crossover = new Crossover($population);
$crossoverOffsprings = $crossover->crossover();

//echo 'Crossover offsprings:<br>';
//print_r($crossoverOffsprings);

echo '<p></p>';
//(new Mutation($population))->mutation();
$mutation = new Mutation($population);
if ($mutation->mutation()){
    $mutationOffsprings = $mutation->mutation();
    //echo 'Mutation offspring<br>';
    //print_r($mutationOffsprings);
    //echo '<p></p>';
    foreach ($mutationOffsprings as $mutationOffspring){
        $crossoverOffsprings[] = $mutationOffspring;
    }
}
//echo 'Mutation offsprings <br>';
//print_r($crossoverOffsprings);
$fitness->fitnessEvaluation($crossoverOffsprings);

$selection = new Selection($population, $crossoverOffsprings);
$selection->selectingIndividus();


// $individu = new Individu;
// print_r($individu->createRandomIndividu());
```

Ln 37, Col 32 Spaces: 4 UTF-8 CRLF PHP

Products.txt

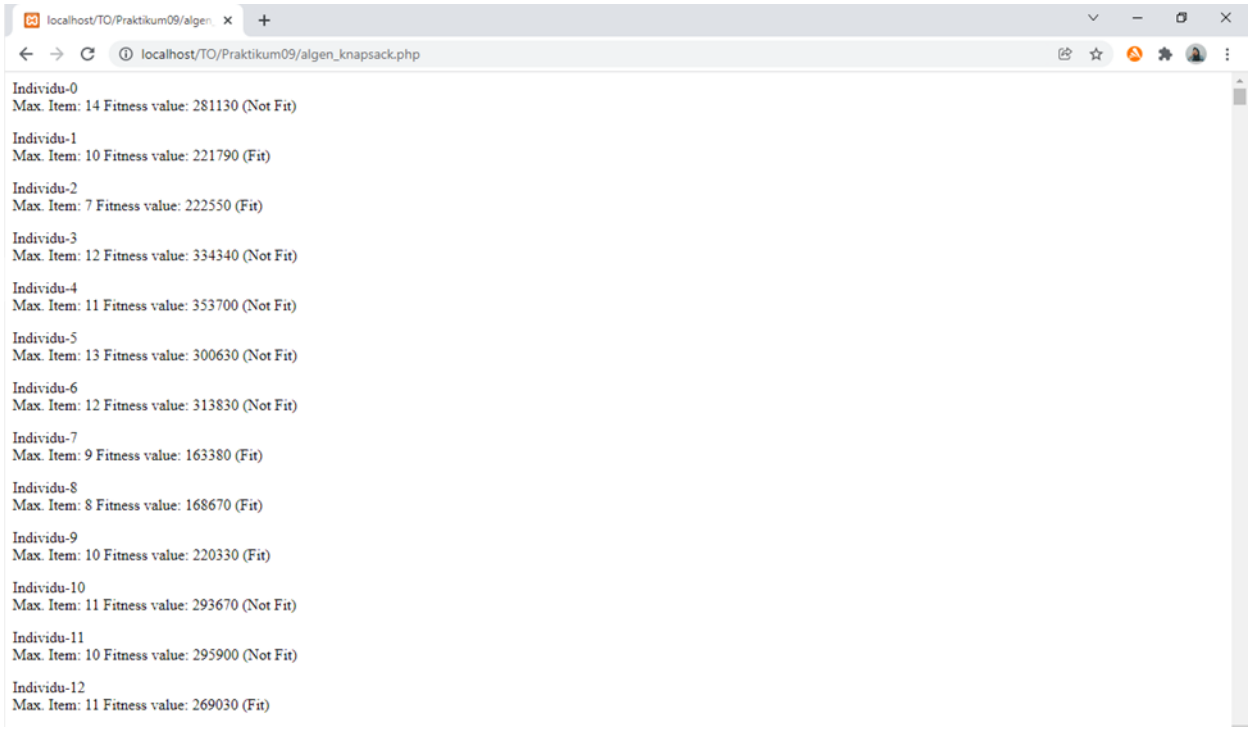
algen_knapsack.php products.txt

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

```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text

Output :



Iadiz-idu-13
Mr Itao 10 bit)

MamNoo:9Fini=sialue:226750(Fit)

Mr Item: 14 Ffiness vine: 244880 (Fit)

Individu-16
Max. Item: 10 Fitness value: 218030 (Fit)

Indivñdu-l7
Mr Item- 11 Fitness v-alue 198530 (Fit)

Max. Item: 11 Fitness value: 183680 (Fit)

Mmh

ladizñdu—20

Ma lter0 14 -alue: 249330 t}

bdhSdu21
Marlnro:11Wmrssiaue:267590(Fñ)

Indiiudo-22
Mr Item: 15 Fitness vine: 270830 (Fit)

Individu-23
Max. Item: 10 Fitness value: 207990 (Fit)

Indivñdu-24
Mr nem 14 378980 (Not Fit)

Ioduadu-25
ld N : 10Fi oss Auo:333500/#Nl)

Iodiadu-26
lvlam heco: 6Fime* clue: 135450(Fit

Individu-27
Mr Item 7 Fitness 225830 (Fit)

Individu-28
Mr Item: 14 Fitness vine: 374720 (Not Fit)

Individu-29
Mr Item 12 F 327970 (Not Fit)

Best value:270830 Residual:9170 Found

Individu-0

Mr Item- I l Fitness v-alue - 248690 (Fit)

Mr Item l3 F 254230 (Fit)

Individu-2
Mr Item: 8 Fitness vine: 293100 (Not Fit)

Mr Item l3 F 210580 (Fit)

Individu-4
Mr Item: 14 Fitness vine: 370140 (Not Fit)

Individu-5
Mr hem 12 F 245330 (Fit)

Mr Item 14 F 357530 (Not Fit)

Individu-7

Mr Item: 13 Fitness vine: 224230 (F Individu-8
Mr Item: 13 clue: 342730 (Not Fit)

Individu-9
Mr limn 13 Fitness false 252230 (Fit)

l d› du1 0

Mr mm: 10 ›Ww: 192280 {bit)

Individu 11
Mr Itmn 13 Fihiess ›al»e 252230 (Fit)

Mr limn 10 Fitness false 204470 (Fit)

ladi du 13
Mr Iteaz: 12 x-aI»e: 243330 bit}

Max. Item: 12 Fitness value: 314220 (Not Fit)

lodi du 15
Mmrleoo:13Wm:sinus:260B0[Fñ)

Individu-16
Mr limn II Furness value 352800 (Not Fit)

Mr Iteaz: 13 x-aI»e: 224230 bit}

Individu 18
Mr Item- 10 Fitness true- 278780 (Fit)

Max. Item: 14 Fitness value: 233130 (Fit)

ladi du—20
Mr Iteaz: 9 x-aI»e: 274730 bit}

Individu-21
Mr Item: 14 clue: 233130 (Fit)

Individu-22
Mr limn l2 Fitness false 220730 (Fit)

1ndimdu-23
Mr Item: 16 clue: 305280 (Not Fit)

ladi du—24
Mr Item- 10 Fitness true- 231680 (Fit)

Individu-25
Mr limn 14 Fitness false 267480 (Fit)

ladi du—26
Mr Iteaz: 14 x-aI»e: 291230 jot Fit}

Individu-27
Max. Item: 11 Fitness value: 188430 (Fit)

ladi du—28
Mr Item: 10 clue: 168430 (Fit)

ladi du—29
Max. Item: 15 Fitness value: 296380 (Not Fit)

Mr Iteaz: 12 x-aI»e: 229280 /it}

Individu-31
Mr Item: 14 clue: 233130 (Fit)

Mr limn 14 Fitness false 297330 (Not Fit)

Individu-33
Mr Item: 14 clue: 233130 (Fit)

Individu-8
Mr Item: 13 clue: 342730 (Not Fit)

Individu-9
Mr limn 13 Fitness false 252230 (Fit)

l d› du1 0

Mr mm: 10 ›Ww: 192280 {bit)

Individu 11
Mr Itmn 13 Fihiess ›al»e 252230 (Fit)

Mr limn 10 Fitness false 204470 (Fit)

ladi du 13
Mr Iteaz: 12 x-aI»e: 243330 bit}

Max. Item: 12 Fitness value: 314220 (Not Fit)

lodi du 15
Mmrleoo:13Wm:sinus:260B0[Fñ)

Individu-16
Mr limn II Furness value 352800 (Not Fit)

Mr Iteaz: 13 x-aI»e: 224230 bit}

Individu 18
Mr Item- 10 Fitness true- 278780 (Fit)

Max. Item: 14 Fitness value: 233130 (Fit)

ladi du—20
Mr Iteaz: 9 x-aI»e: 274730 bit}

Individu-21
Mr Item: 14 clue: 233130 (Fit)

Individu-22
Mr limn l2 Fitness false 220730 (Fit)

1ndimdu-23
Mr Item: 16 clue: 305280 (Not Fit)

ladi du—24
Mr Item- 10 Fitness true- 231680 (Fit)

Individu-25
Mr limn 14 Fitness false 267480 (Fit)

ladi du—26
Mr Iteaz: 14 x-aI»e: 291230 jot Fit}

Individu-27
Max. Item: 11 Fitness value: 188430 (Fit)

ladi du—28
Mr Item: 10 clue: 168430 (Fit)

ladi du—29
Max. Item: 15 Fitness value: 296380 (Not Fit)

Mr Iteaz: 12 x-aI»e: 229280 /it}

Individu-31
Mr Item: 14 clue: 233130 (Fit)

Mr limn 14 Fitness false 297330 (Not Fit)

Individu-33
Mr Item: 14 clue: 233130 (Fit)

India-idu-33
Max Item: I^ Fimess blue: 233. 130 bit)

Max Item 1. 0 Fitness 140 (Not Fit)

India-idu-35
Max Item: 15 Fimess blue: 24.83.80 bit)

Max Item 1. 2 Fitness 27.

(Fit)

India-idu-37
Max Item: 12 Fimess blue: 217330 bit)

Max Item 13 Fitness 83.730 (Not Fit)

India-idu-39
Max Item: 15 Fimess blue: 2763. 80 bit)

lndiv+du-J0
Max Item 1. 1 Furness value 3^H-00 jot Fri)

India-idu-4 I
Max Item: 13 Fimess blue: 274230 bit)

lndiv+du-J2
Max Item 7 Fimess 156000 (Fit)

India-idu-43
Max Item: I3 Fimess blue: 2605. 80 bit)

Max Item 1. 0 Fitness 18530 (Not Fit)

India-idu-45
Max Item: 11 Fitness i-alue: 188^ 30 (Fit)

India-idu-46
Max Item: 12 Fimess blue: 3505. jot Fit)

lndi -idu-J7
Max Item I S Fitness 305280 (Not Fit)

Max. Item: 13 Fitness value: 382870 (Not Fit)

India-idu-49
Mr Item 13 Fitness 226230 bit)

lndiv+du-50
Max Item 13 Fitness 25J230 (Fit)

India-idu-5 I
Max Item: 11 Fitness i-alue: 2^8690 (Fit)

lndiv+du-52
Max Item 7 Fimess 266200 (Fit)

India-idu-53
Max Item: 10 Fimess blue: 178140 bit)

India-idu-SJ
Max Item 13 Fitness 240 (Not Fit)

Max Item: 9 Fimess blue: 212890 bit)

India-idu-56
Max Item-9 Fimess 11550 jot Fit)

Max Item 1. 2 Fitness 263940

(Fit) India-idu-58
Max Item: I3 Fimess blue: 330630 jot Fit)