CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation

1. Form Input

- *Insert*

Below is a fragment of the program to create a user interaction page that will be used for the insert,

```
1. <form method="post">
```

Explanation for the above program fragment is the form used here is a useful post type to get the value of the input box contents.

```
2. <td>Masukkan banyak data : </td>
3. <td><input type="number" name="inputData" min="1"
   max="12"><input type="number" name="angkamin"
   placeholder="Angka minimum"><input type="number"
   name="angkamax" placeholder="Angka maksimum"> <button
   type="submit" name="input">Proses</button></td>
4. </form>
```

Explanation for the above program fragment is Input first, the the type to be used in this page is the numbers, because with numbers it makes it easier for the program to create a process of appeal values that will be used in the binary tree. The initial name used is "inputData", which will be used to retrieve data in the form of value entered by the user. "**min**" serves to set the low value in the filling box, and "**max**" Serves to set the high value in the filling box. Input second, the the type to be used in this page is the numbers, this function as the minimum value for a random system. The initial named used is "angkamin". "**placeholder**" this function as displays "Angka minimum" in the content box. Input third, the the type to be used in this page is the numbers, this function as the maximum value for a random system. The initial named used is "angkamax". "**placeholder**" this function as displays "Angka maksiimum" in the content box. Button in

this form used to submit all value from user to proceed to the next page for *insert*. The next process is to make the number of boxes for later filled and stored in txt by using php code, that is,

```
5. echo "<form method='post' action='banyakData.php'
   target='_blank'>";
```

Explanation for the above program fragment is a command to create an html form, which uses the post method to retrieve data and process it again at the next process (**action='banyakData.php'**).

```
6.      for($i=0;$i<$inputData;$i++)
7.      {
8.          $random=rand($angkamin,$angkamax);
9.          echo "$i <input type='number'
   name='data[$i]' value='$random' required><br><br>";
10.     }
```

Explanation for the above program fragment is a command to create a random system (**$random**) and display it in the html input contents box automatically whose data is extracted from a content box named "**inputData**". To process this random system take a data that has been input user earlier by using variable **$angkamin** for the lowest value and **$angkamax** for the highest value by using the POST command in php. The type of data content box used here is the number (**type='number'**). The name of data content box used here is **data[$i]**, means that the name of this data follows the amount of data to be printed in the for command. **Value** used here serves to print the results of a random system. **Required** is a command that tells the contents of the box can not be empty and must be filled.

```
11.             echo "<button type='submit'
   name='ok'>Ok</button>";
12.             echo "</form>";
```

Explanation for the above program fragment is to create a button button to proceed to the next page, and display all of these forms. Here is a program to store in txt, that is;

```
13.     $open=fopen("data.txt","w");
```

Explanation for the above program fragment is **$open** serves to call data.txt and write its contents.

```
14.             for($i=0;$i<count($_POST['data']);$i++)
15.             {
16.                     //echo "Memasukkan Data $i ".
  $_POST['data'][$i]."<br>";
17.                     $enter="";
18.                     if($i!=0)
19.                     {$enter="\n";}
20.                     $data=$enter.$_POST['data'][$i];
```

Explanation for the above program fragment is fill data.txt with data that has been inputted from the user.

```
21.                     if(fwrite($open,$data))
22.                     {
23.                             print
  "<script>window.alert('Data masuk $i');</script>";
24.                     }
```

Explanation for the above program fragment is generates a message which means that data 1, data 2, data 3, ... has been entered into data.txt.

```
25.             }
26.             fclose($open);
```

Explanation for the above program fragment is closing data.txt that has been successfully processed.

- *Search*

Below is a fragment of the program to create a user interaction page that will be used for the search,

```
27.     <form method="post" action="TestDriveCari.php"
  target="_blank">
```

Explanation for the above program fragment is the form used here is a useful "**post**" type to get the value of the input box contents, "**action**" in this form is used for forward to the process page, "**target=_blank**" in this form is used for direct a new tab in the browser.

```
28.             <td>Mencari data : </td>
29.             <td><input type="number" name="cariData">
30.                     <button
  type="submit">Cari</button></td>
```

```
31.           </form>
```

Explanation for the above program fragment is input in this form, has named ”cariData” and in entering data in the form of numbers. Button in this form used to submit all value from user to proceed to the next page for *search*.

➢ *Delete*

Below is a fragment of the program to create a user interaction page that will be used for the delete,

```
32.     <form method="post" action="TestDriveHapus.php"
   target="_blank">
```

Explanation for the above program fragment is the form used here is a useful “**post**” type to get the value of the input box contents, “**action**” in this form is used for forward to the process page, “**target=_blank**” in this form is used for direct a new tab in the browser.

```
33.     <td>Hapus data : </td>
34.     <td><input type="number" name="hapusData">
35.     <button type="submit">Hapus</button></td>
36.     </form>
```

Explanation for the above program fragment is input in this form, has named ”hapusData” and in entering data in the form of numbers. Button in this form used to submit all value from user to proceed to the next page for *delete*.

2. Binary Tree

To create a binary tree function it requires a container to store data. Below is a fragment of the program,

```
37.     function Node($item)
38.     {
39.           $this->value=$item;
40.           $this->right=null;
41.           $this->left=null;
42.     }
```

Variables that will be used this program includes the **item** to receive data the **value** to store a data to be processed, **left** to store the data relation left, and **right** to store data relation right. Next will be discussed about the process of insert, search, and delete, that is;

- *Insert*

  Below is a fragment of the program to create a user interaction page that will be used for the process to add data,

```
43.     public function insert($item)
44.           {
45.                 $node=new Node($item);
46.
47.                 if($this->kosong())
48.                 {
49.                       $this->root=$node;
50.                 }
```

Explanation for the above program fragment is a condition if empty then created a new Node object to create data (**root**), from **$item** as the recipient of data.

```
51.                 else
52.                 {
53.                       $this->insertNode($node,$this-
  >root);
54.                 }
55.           }
```

Explanation for the above program fragment is a condition if the data storage is no longer empty then it will be ready to be processed to make the relation.

```
56.
57.           function insertNode($node,&$subtree)
58.           {
59.
60.                 if($subtree===null)
61.                 {
62.                       $subtree=$node;
63.                 }
```

Explanation for the above program fragment is a recursive method called from the **insert()** method for the process of making relations. The

above conditions explain about if the state of the relationship is still empty then made new data.

```
64.                  else
65.                  {
66.                        if($node->value > $subtree-
   >value)
67.                        {
68.                              $this->insertNode($node,
   $subtree->right);
69.                        }
70.                        else if($node->value <=
   $subtree->value)
71.                        {
72.                              $this->insertNode($node,
   $subtree->left);
73.                        }
74.
75.                  }
76.            }
```

Explanation for the above program fragment is a condition if the relation is not empty then the process is made for the right relation and the left relation. Create a left relation if the new value data is smaller than the value data from root. Create the right relation if the new value data is greater than the value data from root.

- Search

In the binary tree is also required a search function which will be useful for the delete function. Below there will be a fragment of the program and its explanation of the search function, that is;

```
77.      function cari($target)
78.            {
79.                  $file=file("data.txt");
```

Explanation for the above program fragment is to read data.txt file which will be executed using array. **`$target`** works as a recieve data.

```
80.                  for($i=0;$i<count($file);$i++)
81.                  {
82.                        $value=intval($file[$i]);
83.                        if($target==$value)
84.                        {
```

```
85.                                 print "<p
  style='color:blue;'>Data ke-$i => ".$file[$i]."</p>";
86.                                 print "<p
  style='color:blue;'>Data $target ketemu</p>";
87.
88.                         }
```

Explanation for the above program fragment is a looping process in which it contains the condition if the data entered is equal to the data contained in txt it will display a blue colored message with the text “Data ... ketemu”.

```
89.                         if($target!=$value)
90.                         {
91.                                 print "<p
  style='color:blue;'>Data ke-$i => ".$file[$i]."</p>";
92.                                 print "<p
  style='color:red;'>Data tidak $target ketemu</p>";
93.                         }
94.                 }
95.         }
```

Explanation for the above program fragment is a looping process that contains the condition if the data entered is not the same as the data contained in the txt will display a red colored message with the text “Data ... tidak ketemu”.

- Delete

The delete process described herein is a composite of some conditions of the search process. Below there will be a fragment of the program and its explanation of the delete function, that is;

```
96.     function hapus($target)
97.         {
98.                 $file=file("data.txt");
```

Explanation for the above program fragment is to read data.txt file which will be executed using array. **`$target`** works as a recieve data.

```
99.                 $open=fopen("data.txt","w");
100.                fclose($open);
```

Explanation for the above program fragment is rewrite the edited data in the data.txt file.

```
101.                for($i=0;$i<count($file);$i++)
102.                {
103.                      $value=intval($file[$i]);
104.                      if($target==$value)
105.                      {
106.                            echo "Menghapus data ".
  $file[$i]."<br>";
107.                            $file[$i] = "";
108.                      }
```

Explanation for the above program fragment is a looping process that has a condition if data from $ target equal to data from txt then will do delete data. Data in txt in this process will be used as integer number (**$value**) to be processed and to delete it then required variable which contains empty data (**$file[$i] = ""**) for use in process delete.

```
109.                      file_put_contents("data.txt",
  $file[$i],FILE_APPEND);
110.                }
111.          }
```

Explanation for the above program fragment is a command that serves to delete data by overwriting data with empty data. The rules required herein include the txt file to be executed, the contents of the new data to be processed, and a command from php to delete the selected data (**FILE_APPEND**).

3. Visualization

In the visualization of the required objects such as circles, lines, etc. Before creating an object must be a canvas, below is a program to create canvas, that is

```
112.     function Canvas($w,$h)
113.           {
114.                $this->img=imagecreate($this->w=$w,
  $this->h=$h);
```

Explanation for the above program fragment is the canvas will appear on the output of a PNG image with a white background. The structure required to form this canvas is width to set the width of the canvas, and height to set the height of the canvas.

```
115.                $bg=imagecolorallocate($this-
  >img,255,255,255);
116.          }
```

Explanation for the above program fragment is to fill white on the background of the canvas. To make a white color then set R, G, B to 255. Below will be given a fragment of programs and explanations, that is;

- Creating a circle for a node

In the GD library there is a command to print a circle in which there is already a color that is imagefilledellipse.

```
117.     function Lingkaran($x,$y,$size1,$size2)
118.          {
119.                $htm=imagecolorallocate($this-
  >img,0,0,0);
```

Explanation for the above program fragment is to give the color black (**$htm**) on the color of the circle.

```
120.                imagefilledellipse($this->img,$this-
  >x=$x,$this->y=$y,$this->size1=$size1,$this-
  >size2=$size2,$htm);
121.     }
```

Explanation for the above program fragment is to create a circle object. The structure required in this program is calling the canvas variable that has been created, x coordinate layout, y coordinate layout, size for width of circle, size for circle height, and color for circle.

- Creating a connecting line for a node

In the GD library there is a command to print a line that is imageline.

```
122.     function Garis($xawal,$yawal,$xakhir,$yakhir)
123.          {
124.                $htm=imagecolorallocate($this-
  >img,0,0,0);
```

Explanation for the above program fragment is to create a black color that will later be used for line color.

```
125.                imageline($this->img,$this-
  >xawal=$xawal,$this->yawal=$yawal,$this-
  >xakhir=$xakhir,$this->yakhir=$yakhir,$htm);
126.     }
```

Explanation for the above program fragment is a command to create a connecting line. Structure used in this program that is calling the canvas variable that has been created, set the top x coordinate layout, set the top y coordinate layout, set the bottom x coordinate layout, Set the bottom y coordinate layout, and gives the color that has been made.

- Create a caption for a node

In the GD library there is a command to print a caption that is imagestring.

```
127.      function String($fsize,$xstring,$ystring,$isi)
128.           {
129.                $pth=imagecolorallocate($this-
  >img,255,255,255);
```

Explanation for the above program fragment is create a white color that will later be used for caption color.

```
130.                imagestring($this->img,$this-
  >fsize=$fsize,$this->xstring=$xstring,$this-
  >ystring=$ystring,$this->isi=$isi,$pth);
131.           }
```

Explanation for the above program fragment is a command to create a caption. Structure used in this program that is calling the canvas variable that has been created, set the font size, set x coordinate, set y coordinate, set the contents to be displayed, and set to give white color that has been made.

- Create a coordinate

To set the object layout then required coordinates by using array() in php. Below is a sample fragment to set the coordinates, that is;

```
132.    $this->arRoot=array($this->w/2,50);
```

Explanation for the above program fragment is to set the x coordinates in the middle (**root**) then the width of the set canvas is divided by 2, then to set the fill height as desired for example 50. Example to create the left coordinate, that is;

```
133.    $this->arKiri1=array($this->arRoot[0]-180,$this-
  >arRoot[1]+100);
```

Explanation for the above program fragment is to set the coordinates of the left x then the required coordinate x of the middle of the (root) that has been set is reduced by 180 or according to the desire to be left, then to set the coordinate y left then the required coordinate y of the center was (root) added 100 Or according to the desire to be in the left position. Neither for set the right relation whose x and y coordinates are added so that it is on the right. Example to create the right coordinate, that is;

```
134.    $this->arKanan1=array($this-
  >arRoot[0]+180,$this->arRoot[1]+100);
```

5.2 Testing

After all the program is formed then what should be done next is the testing phase of the program. Below will explain about the output display of the program described above, that is

1. Experiment *insert* program

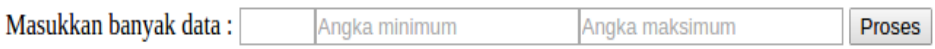


Illustration 5.1: A page for insert data input

Explanation of the picture above is the first box of its function to make how much data will be inputted with a value of only max 12, because in this program only made as much as 8 level, as shown below

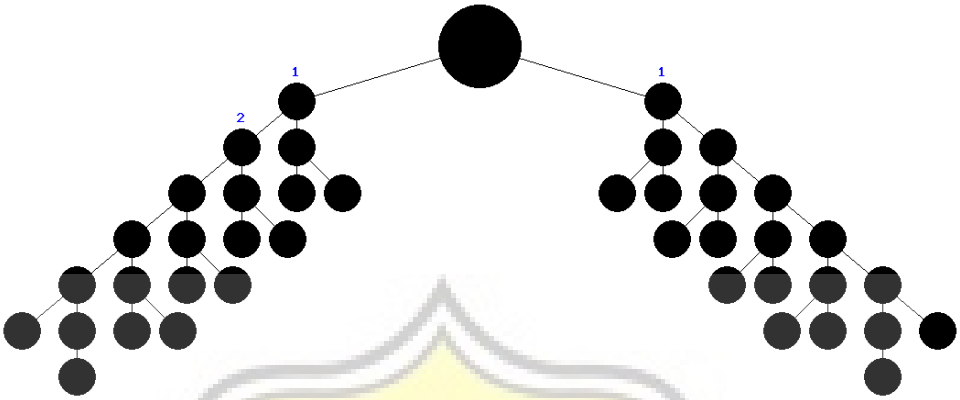
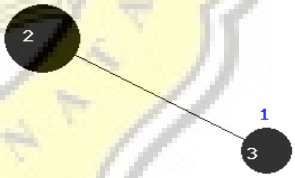
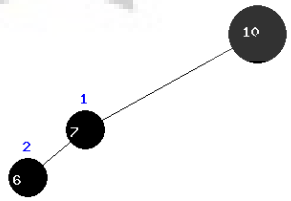
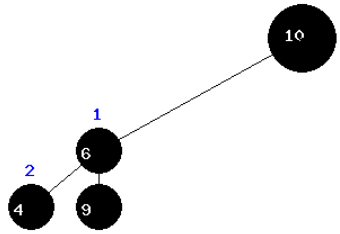


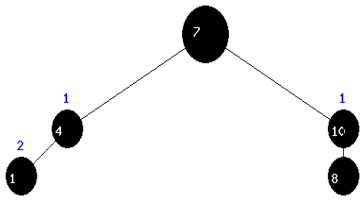
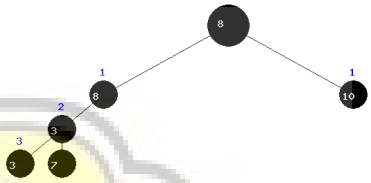
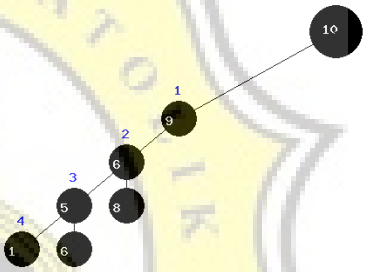
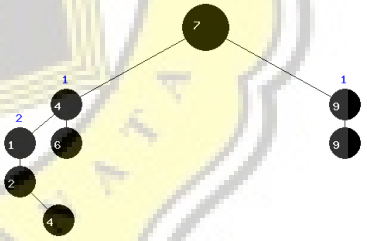
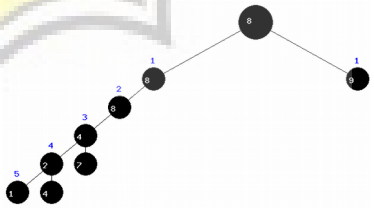
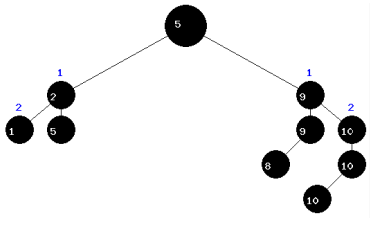
Illustration 5.2: Binary tree in 8 levels

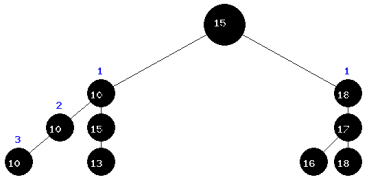
The second and third box function to run the random value by filling the minimum and maximum values, before pressing the button. Below is an experiment of an insert program,

Table 5.1: Experiment insert program

Test	Many data entered	Contents of the data	Results
1	2	2, 3	
2	3	10, 7, 6	

3	4	10, 6, 4, 9	

4	5	7, 10, 8, 4, 1	
5	6	8, 8, 3, 7, 10, 3	
6	7	10, 9, 6, 5, 1, 6, 8	
7	8	7, 4, 6, 9, 1, 9, 2, 4	
8	9	8, 9, 8, 8, 4, 2, 7, 4, 1	
9	10	5, 9, 2, 9, 5, 10, 8, 10, 10, 1	

10	11	15, 18, 10, 17, 15, 10, 13, 10, 18, 18, 16	

Explanation of the above experiment is by entering a lot of data to be input then the program will give the contents box automatically in accordance with the amount of data that has been inputted earlier that has been equipped with a random system to make each random box content that can also be edited its contents. After that, by pressing the ok button it will automatically create a binary tree visualization.

2. Experiment search program

Illustration 5.3: Page for search

The workings of this page is simply to fill in the form of data values in the boxes that have been provided.

Table 5.2: Experiment search program

Test	Many data entered	Contents of the data	Data searched	Result

1	3	5, 4, 3	5	Data ke-0 => 5 Data 5 ketemu --- Data ke-1 => 4 Data tidak 5 ketemu --- Data ke-2 => 3 Data tidak 5 ketemu
2	3	5, 4, 3	4	Data ke-0 => 5 Data tidak 4 ketemu --- Data ke-1 => 4 Data 4 ketemu --- Data ke-2 => 3 Data tidak 4 ketemu

3	3	5, 4, 3	3	Data ke-0 => 5 Data tidak 3 ketemu Data ke-1 => 4 Data tidak 3 ketemu Data ke-2 => 3 Data 3 ketemu
4	4	8, 7, 9, 11	7	Data ke-0 => 8 Data tidak 7 ketemu Data ke-1 => 7 Data 7 ketemu Data ke-2 => 9 Data tidak 7 ketemu Data ke-3 => 11 Data tidak 7 ketemu

5	4	8, 7, 9, 11	9	Data ke-0 => 8 Data tidak 9 ketemu Data ke-1 => 7 Data tidak 9 ketemu Data ke-2 => 9 Data 9 ketemu Data ke-3 => 11 Data tidak 9 ketemu
6	4	8, 7, 9, 11	11	Data ke-0 => 8 Data tidak 11 ketemu Data ke-1 => 7 Data tidak 11 ketemu Data ke-2 => 9 Data tidak 11 ketemu Data ke-3 => 11 Data 11 ketemu

7	5	13, 11, 10, 15, 14	10	Data ke-0 => 13 Data tidak 10 ketemu Data ke-1 => 11 Data tidak 10 ketemu Data ke-2 => 10 Data 10 ketemu Data ke-3 => 15 Data tidak 10 ketemu Data ke-4 => 14 Data tidak 10 ketemu
8	5	13, 11, 10, 15, 14	15	Data ke-0 => 13 Data tidak 15 ketemu Data ke-1 => 11 Data tidak 15 ketemu Data ke-2 => 10 Data tidak 15 ketemu Data ke-3 => 15 Data 15 ketemu Data ke-4 => 14 Data tidak 15 ketemu

9	6	24, 25, 22, 20, 21,23	22	Data ke-0 => 24 Data tidak 22 ketemu --- Data ke-1 => 25 Data tidak 22 ketemu --- Data ke-2 => 22 Data 22 ketemu --- Data ke-3 => 20 Data tidak 22 ketemu --- Data ke-4 => 21 Data tidak 22 ketemu --- Data ke-5 => 23 Data tidak 22 ketemu ---

10	6	24, 25, 22, 20, 21,23	21	Data ke-0 => 24 Data tidak 21 ketemu Data ke-1 => 25 Data tidak 21 ketemu Data ke-2 => 22 Data tidak 21 ketemu Data ke-3 => 20 Data tidak 21 ketemu Data ke-4 => 21 Data 21 ketemu Data ke-5 => 23 Data tidak 21 ketemu

Explanation of the above experiment is after displaying the binary tree visualization, to be able to run the program search according to the above procedure then obtained the results as the table above. In this program, the search data that has been input from the user will be compared with the data content in txt to form the binary tree. If the data from the user is the same as the content data txt then there will be 'ketemu' text in blue color, if not the same then there will be 'tidak ketemu' with red color.

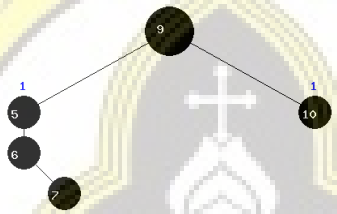
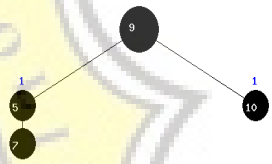
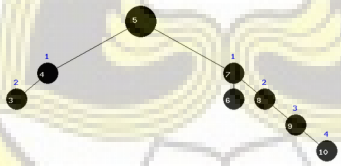
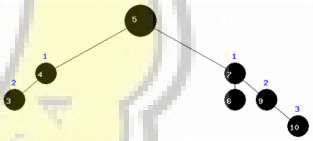
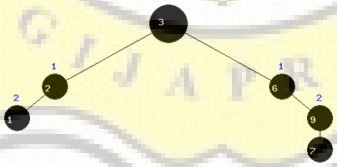
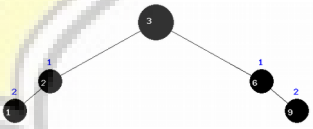
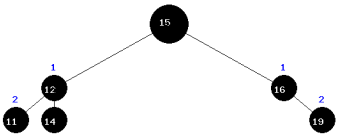
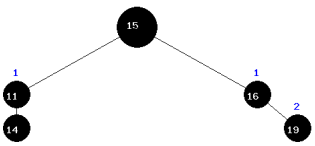
3. Experiment delete program

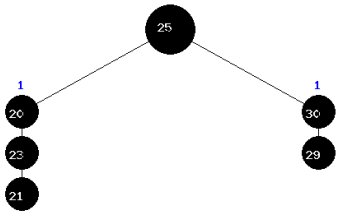
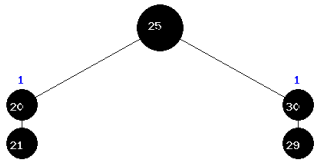
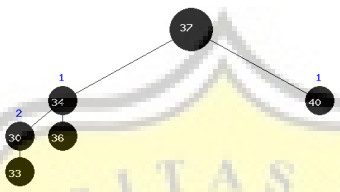
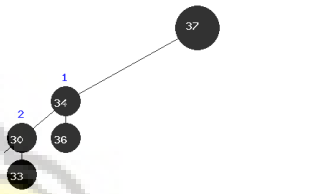
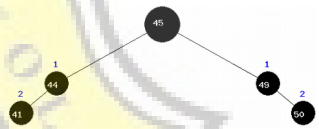
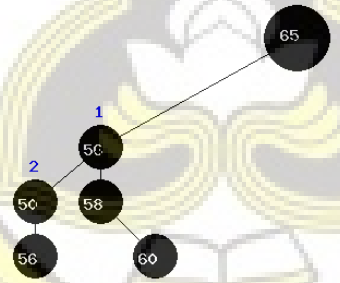
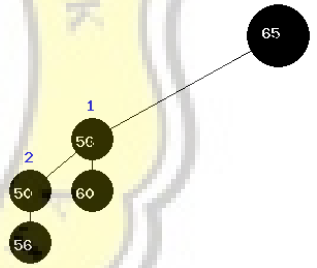
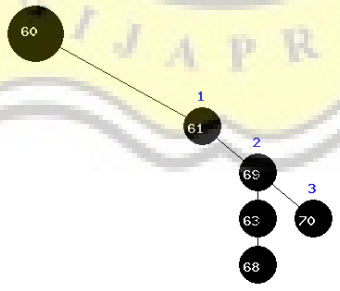
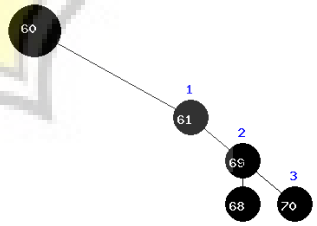
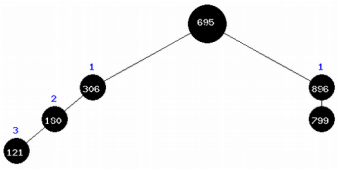
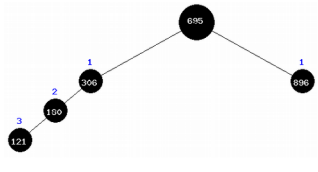
Page for *delete* data input from user

Illustration 5.4: Page for delete

The workings of this page is very simple that is by filling the data numbers to do delete by user, after it will appear the message is successful, and if you want the image then select the image page and refresh.

Table 5.3: Experiments delete program

Test	Content of the data	Insert Result	Deleted data	Delete Result
1	9, 10, 5, 6, 7		6	
2	5, 7, 4, 3, 8, 9, 6, 10		8	
3	3, 6, 9, 7, 2, 1		7	
4	15, 16, 12, 11, 19, 14		12	

5	25, 30, 20, 23, 21, 29		23	
6	37, 34, 30, 33, 36, 40		40	
7	45, 49, 44, 47, 50, 41		47	
8	65, 56, 58, 50, 60, 56		58	
9	60, 61, 69, 63, 70, 68		63	
10	695, 896, 306, 180, 799, 121		799	

Explanation of the above experiment is in this program if the data is successfully deleted message will appear 'menghapus data ...' if not find, no messages will appear or empty, for the results of the image by refreshing browser the result of the binary tree structure page that has been created in the insert program, then there will be a missing node.

All the experiments already described and included are the result of the success of the program process. If there is an error or not successful then the image will not appear.