CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

Shell and exchange sort visualization program is very helpful in the process of sorting the array. With the visualization in the form of the program, the user can understand how the sorting work. The process displayed on the display is not only a direct answer, but also shows the process of looping from the beginning to the end of the process. Below are some important coding supporting the course of the program along with its function.

1. **Set_elements**

   The following code is coding for creating boxes.

   ```
   function set_elements(n)
   {
   var result = $('#result');
   var i;
   result.html('');
   result.css('position', 'relative');
   result.css('margin', '100px 0 0 0');
   result.css('height', '100px');
   for (i = 0; i < n; i = i + 1)
   {
   var x = 50 * i;
   item.attr('id', 'el_' + i);
   item.css('position', 'absolute');
   item.css('width', '30px');
   item.css('height', '30px');
   item.css('text-align', 'center');
   item.css('line-height', '30px');
   item.css('border', '2px solid #0000FF');
   item.css('left', x);
   result.append(item);
   }
   }
   ``

   Coding on line 14 and 15 is used to determine the length and width of boxes to be sorted. Line 13 to adjust the position of the grid on the screen. Line 18 gives color to box thickness. This visualization will
show more than 1 box, then there will be a distance on box 1 with another.
Coding on line 11 is to make the distance between boxes.

2. Function exchange sort

The following code is to code to run the exchange sort algorithm.

```
for(var j=1; j < (n-i); j++)
{
    if(listNumber[j-1] > listNumber[j])
    {
        var temp = listNumber[j-1];
        listNumber[j-1] = listNumber[j];
        val2 = listNumber[j];
        listNumber[j] = temp;
    }
}
```

In the above coding contains the settlement using exchange sort algorithm. In line 3 contains the command if the previous number is larger than the current number, then the two numbers will be exchanged. Line 5 describes the contents of the box in the previous index, while the 8th row contains the boxes in the current index. On line 6 contains if both numbers are equal then the position will be fixed or not exchanged.

3. Function shell sort

The following code is to code to run the shell sort algorithm.

```
while (h <= len / 3)
{
    h = h * 3 + 1;
    console.log('nilai interval' + h);
}
```

In the above coding contains the settlement using shell sort algorithm. In line 3 contains the command if the previous number is larger than the current number, then the two numbers will be exchanged. Line 5 describes the contents of the box in the previous index, while the 8th row contains the boxes in the current index. On line 6 contains if both numbers are equal then the position will be fixed or not exchanged.
15.     {
16.         listNumber[inner] = listNumber[inner - h];
17.         val2 = listNumber[inner];
18.         inner -= h;
19.     }
20.     listNumber[inner] = temp;
21.     h = (h - 1) / 3; // decrease h
22.     i++;
23.}

The above encoding contains the completion using the shell sort algorithm. Line 3 contains an interval value search formula. The program the value of the interval used 4. At line 10, temp is used to display the value of the index (outer). The lines 13 to 22 contain the formula of the shell sort algorithm.

5.2 Testing

Here is a visualization view of the process of sorting numbers by using the shell and exchange sort method.

5.2.1 Exchange sort process

Illustration 5.1: Main page view
1. Main page views on the shell and exchange sort visualization program. There are various buttons are insert, random, clear, shell, and exchange sort.

2. The insert button is used to call and display the on-screen box according to the number of inserts. For example, 8 boxes insert will appear 8 boxes. Inside the box will appear a random number of between 1 – 99.
3. The random button is used to randomize the number box until the number box position changes. In this program the position of numbers in the random setting as much as the number of boxes inputted. The number box will exchange places by shifting. For animation movements on a number box with each other shift.

4. The result of an exchange sort of Iteration starts from 0 and a pink box is a box indicating an iteration process.
Illustration 5.5: Iteration / looping 1
Illustration 5.6: Iteration / looping 2

Illustration 5.7: Iteration / looping 3
5. The process of sorting is completed until the 5 iteration.

Illustration 5.8: Iteration / looping 4

Illustration 5.9: Iteration / looping 5
### 5.2.1 Shell sort process

1. The pink box in the image above indicates the interval distance. Box 63 as index to 0 while box 54 as index to 4. When the first number box in index is bigger than second index then move it. As marked on the pink box. Box 63 will move to index 4 while box 54 will instead move to index 0. After the process will experience looping again from the number box in index 1 to the number box on index 5 and compare each other until the looping finish.
2. After no more box number at interval 4. Then enter interval 1, almost equal to interval 4 difference in index. At interval 1 the index is between 0 and index 1. Results from shell sort process.