# CHAPTER 5

# IMPLEMENTATION AND TESTING

## 5.1 Implementation

Implementation stage to discuss about the operation of the program to be made. The program has operations such as insert sort, delete head, delete tail and search. Insert sort is the inputted data will be automatically sorted by the program. Delete head is the stored data will be removed from the front. Delete tail is the stored data will be removed from the back and search to search for inputted data.

```
1       public void addSort(String val){
2             Node newNode = new Node(val);
3               if (head == null){
4                   head = newNode;
5                   tail = newNode;
6               }else{
7                   Node curr = head;
8                   int val = Integer.valueOf(value);
9                   Log.i("indo",val+"");
10                  Node temp = null;
11                  boolean headBool = false;
12                  while(curr !=null){
13                        if(val>Integer.valueOf(curr.data)){
14                              temp = curr;
15                        }
16                        else if(val<Integer.valueOf(head.data)){
17                              temp = curr;
18                              headBool = true;
19                              break;
20                        }
21                        curr = curr.next;
22                  }
23
24                  if(headBool){
25                        newNode.setNext(head);
26                        head = newNode;
27                  }else{
28                        newNode.setNext(temp.next);
29                        temp.setNext(newNode);
30                  }
31              }
32          }
```

Source code above serves to sort the data that has been inputted into the linked list. The first input data will be the head and the inputted data will then be compared with the existing data to determine the head and tail. If the data is larger than the data head, then the data will be stored as the tail. If the data is smaller than the data head, then the data will be stored as the head.

```
1       public int getPos(String a){
2                 Node temp=head;
3                 Node find=null;
4
5                 int count = 0;
6                 boolean exist = false;
7                 if(temp == null){
8                       System.out.println("Not Found");
9                 }
10                else{
11                while(temp != null){
12                      if(temp.data.equals(a)){
13                            find=temp;
14                            System.out.println("Found");
15                            exist = true;
16                            break;
17                      }
18                      else{
19                            temp = temp.getNext();
20                      }
21                      count++;
22                }
23          }
24          return (exist) ? count : -1;
25      }
```

Source code above serves to find the position of data that has been inputted into the linked list. If the existing data with the data that is input, it will display a notification of data found. Conversely, if not the same, it will display a notification can not be found. Count: -1 serves to create canvas as a sign of data found or not found.

```
1       if(pos != -1){
2       canvas.drawRect(varx+10,vary,panjangx+180,panjangy,
        paint3);
3       canvas.drawLine(linex+140,liney,linepx+140,linepy,
        paint3);
4       Toast.makeText(MainActivity.this,"Position"+String.valueO
        f(pos), Toast.LENGTH_LONG).show();
5       }
6       else{
```

```
7       Toast.makeText(MainActivity.this,"Data   Not   Found",
        Toast.LENGTH_LONG).show();
8       }
```

If pos is not equal -1, it will create a canvas and a notification of the position of the data sought. If pos is equal -1, the data notification will not be found.

```
1       if(count == 0){
2            int varx = 10;
3            int vary = 100;
4            int panjangx = 180;
5            int panjangy = 200;

6            int linex =140;
7            int liney = 100;
8            int linepx = 140;
9            int linepy = 200;

10           int tx = 50;
11           int tpy = 170;

12      getCanvas().drawRect(varx,vary,panjangx,panjangy,
        paint1);
13      getCanvas().drawLine(linex,liney,linepx,linepy, paint4);
14      getCanvas().drawText(temp.data, tx, tpy, paint2);
15      }
16      else{
17      getCanvas().drawLine(startX+180,startY,stopX+260,stopY,
        paint4);
18      getCanvas().drawLine(leftpx+230,leftx,stopX+260,lefty,
        paint4);
19      getCanvas().drawLine(rightpx+230,rightx,stopX+260,righty,
        paint4);
20      getCanvas().drawRect(varx+260,vary,panjangx+430,panjangy,
        paint1);
21      getCanvas().drawLine(linex+390,liney,linepx+390,linepy,
        paint4);
22      getCanvas().drawText(temp.data, tx+300, tpy, paint2);
23      }

24      varx = count * 250;
25      vary = 100;
26      panjangx = count * 250;
27      panjangy = 200;

28      linex = count *250;
29      liney =  100;
30      linepx = count *250;
31      linepy = 200;

32      startX = count *250;
```

```
33      startY = 155;
34      stopX = count * 250;
35      stopY = 155;

36      leftx = 140;
37      lefty = 155;
38      leftpx = count * 250;
39
40      rightx = 172;
41      righty = 155;
42      rightpx = count * 250;

43      tx = count * 250;
44      tpy = 170;
```

code above serves to create a canvas, if the count is equal to 0 then it will make the first canvas. Lines 2-13 to determine the coordinate position of creating the first canvas. Lines 15 and 23 to create boxes, rows 16 and 24 to create lines. Lines 17 and 25 to display the inputted data, line 20-22 to make the arrows. If count is not equal to 0 it will make the next canvas. Line 24-44 to determine the coordinate position of making the next canvas.

```
1       <LinearLayout
2               android:layout_width="match_parent"
3               android:layout_height="60dp"
4               android:orientation="horizontal" >

5                <EditText
6                   android:id="@+id/insert1"
7                   android:layout_width="90dp"
8                   android:layout_height="wrap_content" />
9                <EditText
10                  android:id="@+id/valuesearch"
11                  android:layout_width="100dp"
12                  android:layout_height="wrap_content" />

13               <Button
14                  android:id="@+id/btninsert"
15                  android:layout_width="wrap_content"
16                  android:layout_height="wrap_content"
17                  android:text="Insert Sort" />
18               <Button
19                  android:id="@+id/search"
20                  android:layout_width="wrap_content"
21                  android:layout_height="wrap_content"
22                  android:text="Search"/>

23          </LinearLayout>
```

line 1 and 23 is code to create the layout and determine the position in which the layout has a function to edit text and buttons. Line 5-12 of the program code make text editing function to enter the desired data, line 13-22 to create button.

## 5.2 Testing

The test is performed to ensure quality and also know the strengths and weaknesses contained in the application. There are four experiments operation linked list data structure that is applied in the application as follows:

### 5.2.1 Test Insert Sort

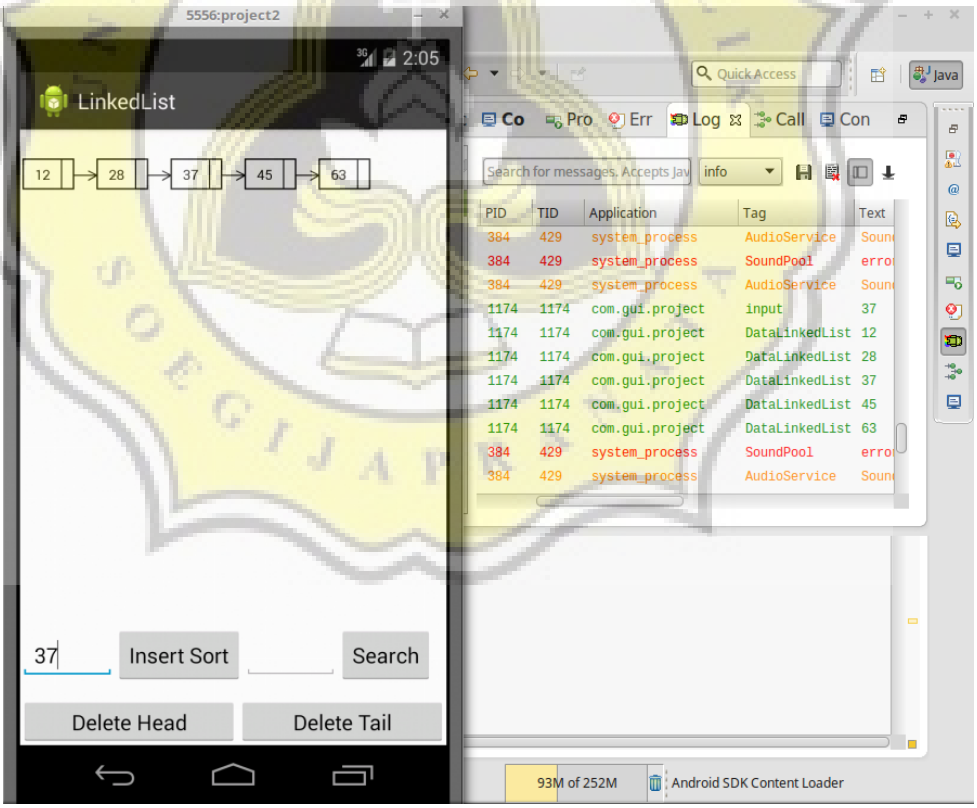


Illustration 5.1: Test Insert Sort

Picture above are the test result of insert sort data inputted. Any data inputted automatically will be sorted. This experiment has displayed a list of data that has been stored in the linked list whether it is in accordance with the data input. Testing the insert sort button is working properly, checking the coordinate position of creating the canvas when the data has been inputted whether it is running properly. In this experiment has successfully performed the operation of insert sort linked list in the form of visualization.

Table 5.1: Test Insert Sort

| No | Insert data | Sorting | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 32 | 12 | 28 | 32 | 37 | 45 | 63 | | | | | | | | | |
| 2 | 75 | 12 | 28 | 32 | 37 | 45 | 63 | 75 | | | | | | | | |
| 3 | 20 | 12 | 20 | 28 | 32 | 37 | 45 | 63 | 75 | | | | | | | |
| 4 | 8 | 8 | 12 | 20 | 28 | 32 | 37 | 45 | 63 | 75 | | | | | | |
| 5 | 52 | 8 | 12 | 20 | 28 | 32 | 37 | 45 | 52 | 63 | 75 | | | | | |
| 6 | 70 | 8 | 12 | 20 | 28 | 32 | 37 | 45 | 52 | 63 | 70 | 75 | | | | |
| 7 | 86 | 8 | 12 | 20 | 28 | 32 | 37 | 45 | 52 | 63 | 70 | 75 | 86 | | | |
| 8 | 30 | 8 | 12 | 20 | 28 | 30 | 32 | 37 | 45 | 52 | 63 | 70 | 75 | 86 | | |
| 9 | 4 | 4 | 8 | 12 | 20 | 28 | 30 | 32 | 37 | 45 | 52 | 63 | 70 | 75 | 86 | |
| 10 | 17 | 4 | 8 | 12 | 17 | 20 | 28 | 30 | 32 | 37 | 45 | 52 | 63 | 70 | 75 | 86 |

Based on the data in the image Illustration 5.2, will be tested insert 10 data. in the testing insert random data, inputted data is automatically sorted by program. This test can be concluded that this program can run well.

### 5.2.2 Test Search

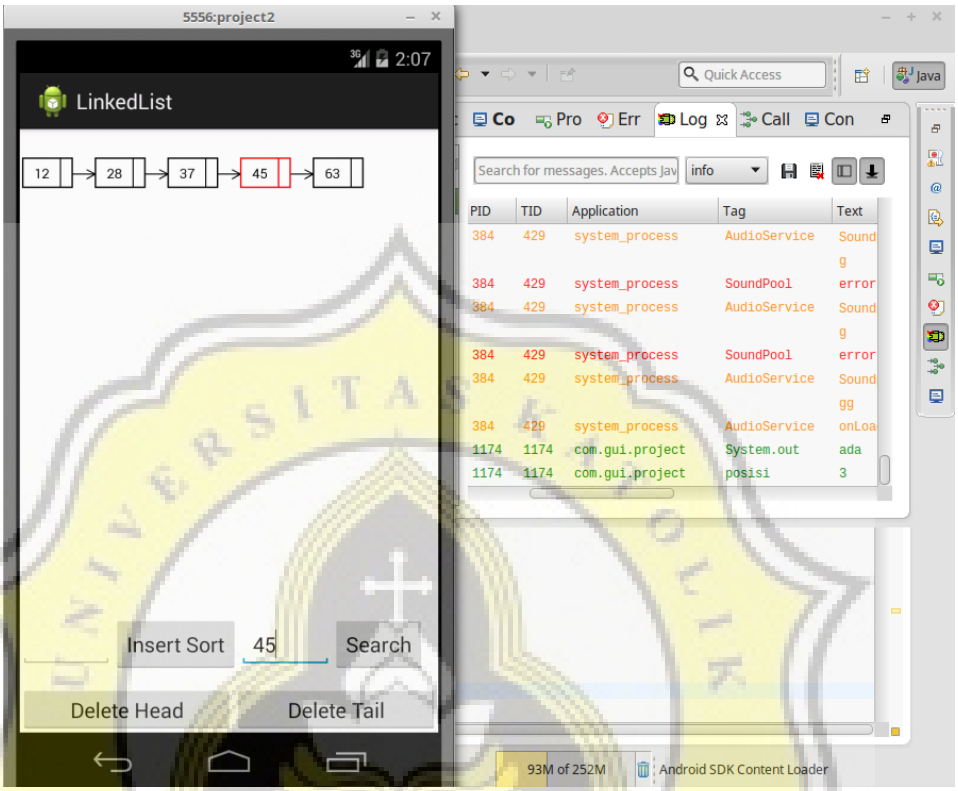


Illustration 5.2: Test Search

Picture above are the test results inputted search data. This experiment to check the search data that inputted whether found or not found. If the data searched has been found, it will display the position of the data. This test is also to check the search button whether it works properly. Check the coordinate position of the canvas when the data is found whether it is correct or false.

Table 5.2: Test Search Data

| No | Input Search data | Data | | | | | Description |
|---|---|---|---|---|---|---|---|
| 1 | 37 | 12 | 28 | 37 | 45 | 63 | Found |
| 2 | 80 | 12 | 28 | 37 | 45 | 63 | Not Found |
| 3 | 12 | 12 | 28 | 37 | 45 | 63 | Found |
| 4 | 7 | 12 | 28 | 37 | 45 | 63 | Not Found |
| 5 | 38 | 12 | 28 | 37 | 45 | 63 | Not Found |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | 45 | 12 | 28 | 37 | 45 | 63 | Found |
| 7 | 78 | 12 | 28 | 37 | 45 | 63 | Not Found |
| 8 | 28 | 12 | 28 | 37 | 45 | 63 | Found |
| 9 | 63 | 12 | 28 | 37 | 45 | 63 | Found |
| 10 | 23 | 12 | 28 | 37 | 45 | 63 | Not Found |
| *Note: 12, 28, 37, 45, 63 is data from Illustration 5.2.* | | | | | | | |

Based on the test table with input 10 data to be searched can be concluded that search data goes well. In the experiments conducted search with the data inputted whether found or not found. If the data is found, it will display the data position with a red border.

### 5.2.3 Test Delete Head

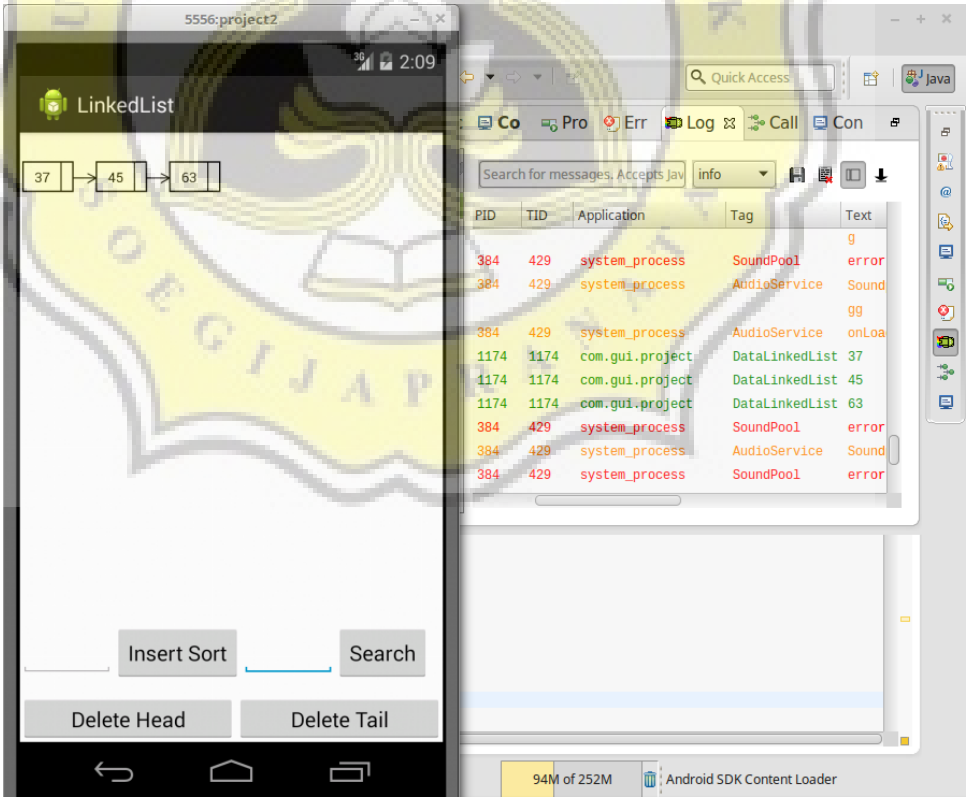


Illustration 5.3: Test Delete Head

Picture above are the test results to remove the head. There are five data already saved from the previous image. From the existing data do testing to remove the head. This test to check to remove the head whether it works properly. Checking whether the stored data has been erased or not. If the data has been erased automatically the displayed data will be reduced from the head. This test is also to check the head clear button whether it is working properly. Checking the coordinator position removes the canvas from the head whether it is right or wrong.

### 5.2.4 Test Delete Tail

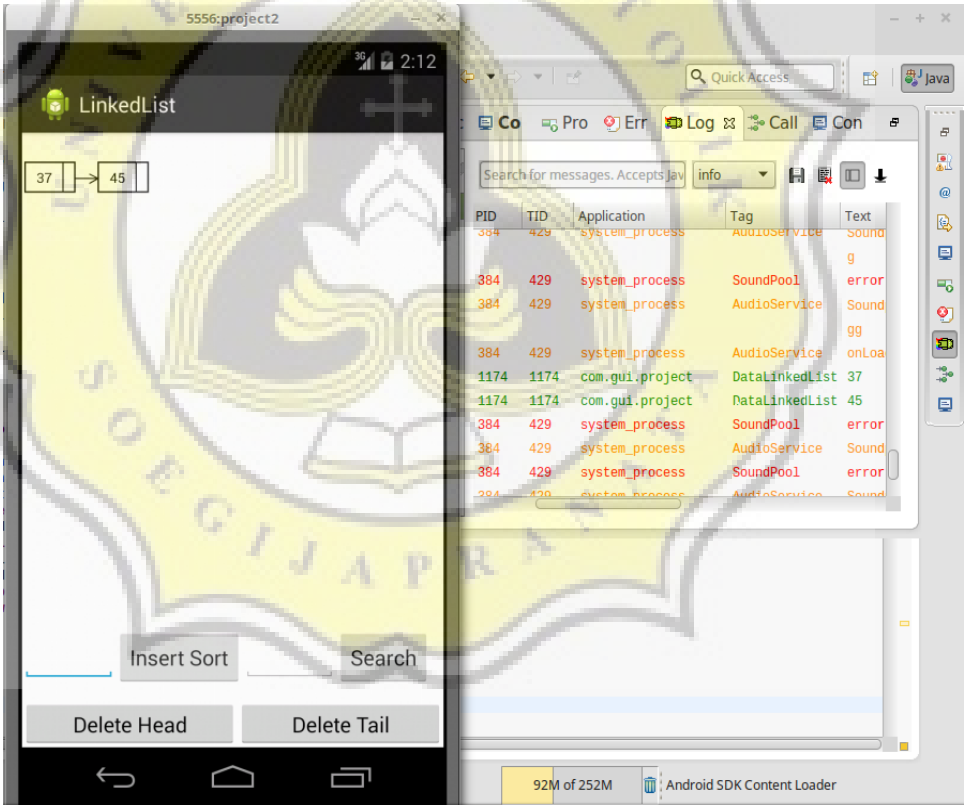


Illustration 5.4: Test Delete Tail

Picture above are the test results to remove the tail. From the previous image, there are three remaining and stored data. From the existing data will be tested remove tail. This test to check the remove tail whether it works properly. checking whether the stored data has been erased or not. If the data has been

deleted automatically the displayed data will be reduced from the tail. This test is also to check the button remove tail whether it is working properly. Checking the coordinate position removes the canvas from the tail whether it is correct or false.