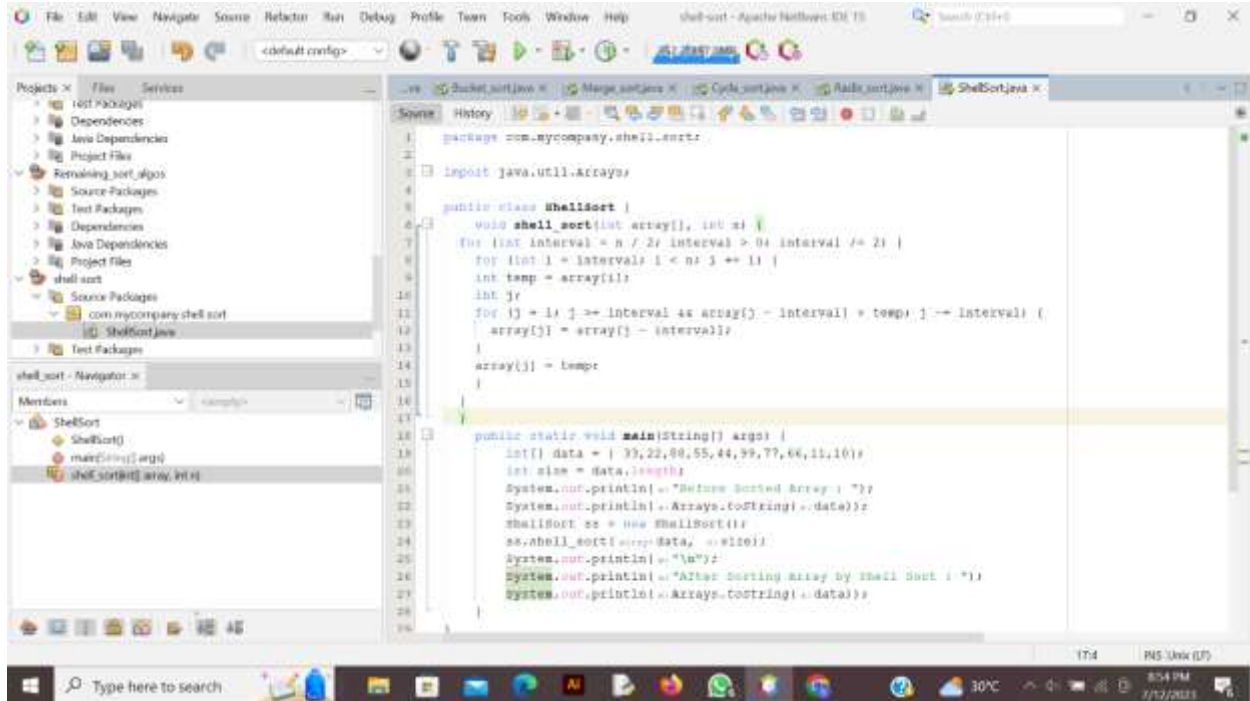


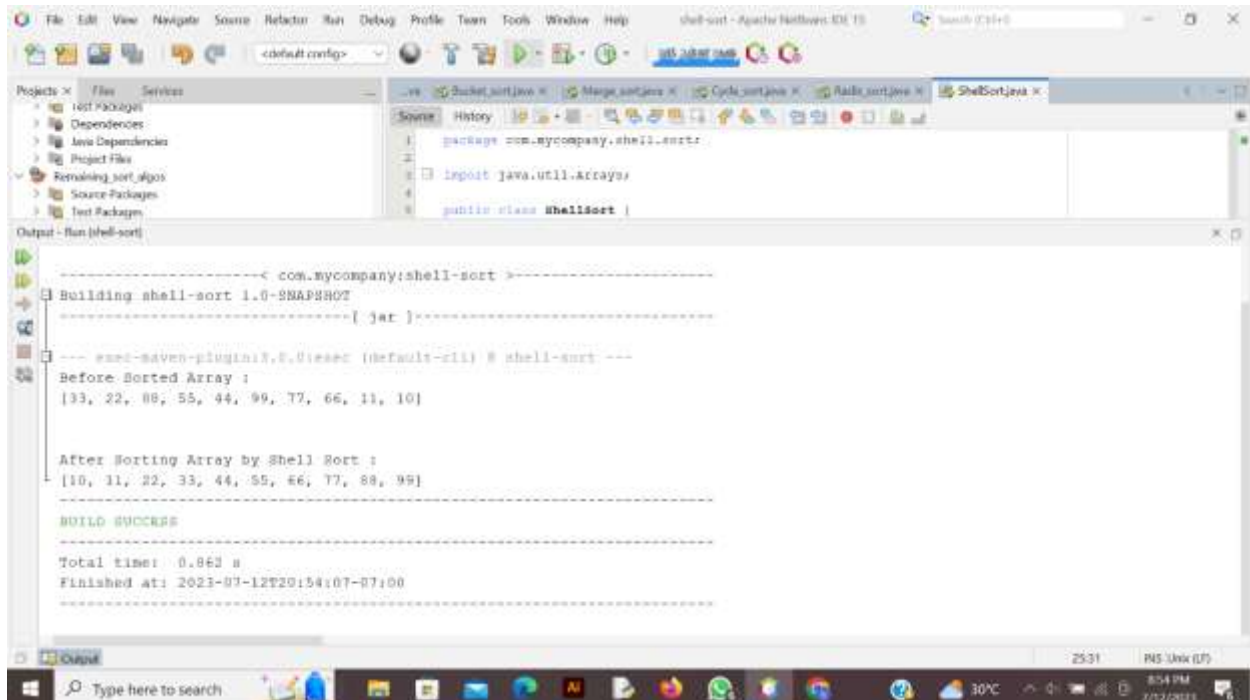
# Today's Task (Tic Tac Toe)

## Shell Sort



The screenshot shows an IDE with the following code in ShellSort.java:

```
1 package com.mycompany.shell.sort;
2
3 import java.util.Arrays;
4
5 public class ShellSort {
6     void shell_sort(int array[], int n) {
7         for (int interval = n / 2; interval > 0; interval /= 2) {
8             for (int i = interval; i < n; i += interval) {
9                 int temp = array[i];
10                int j;
11                for (j = i; j >= interval && array[j - interval] > temp; j -= interval) {
12                    array[j] = array[j - interval];
13                }
14                array[j] = temp;
15            }
16        }
17    }
18
19    public static void main(String[] args) {
20        int[] data = { 33, 22, 88, 55, 44, 99, 77, 66, 11, 10 };
21        int size = data.length;
22        System.out.println("Before Sorted Array : ");
23        System.out.println(Arrays.toString(data));
24        ShellSort ss = new ShellSort();
25        ss.shell_sort(data, size);
26        System.out.println("\n");
27        System.out.println("After Sorting Array by Shell Sort : ");
28        System.out.println(Arrays.toString(data));
29    }
30 }
```



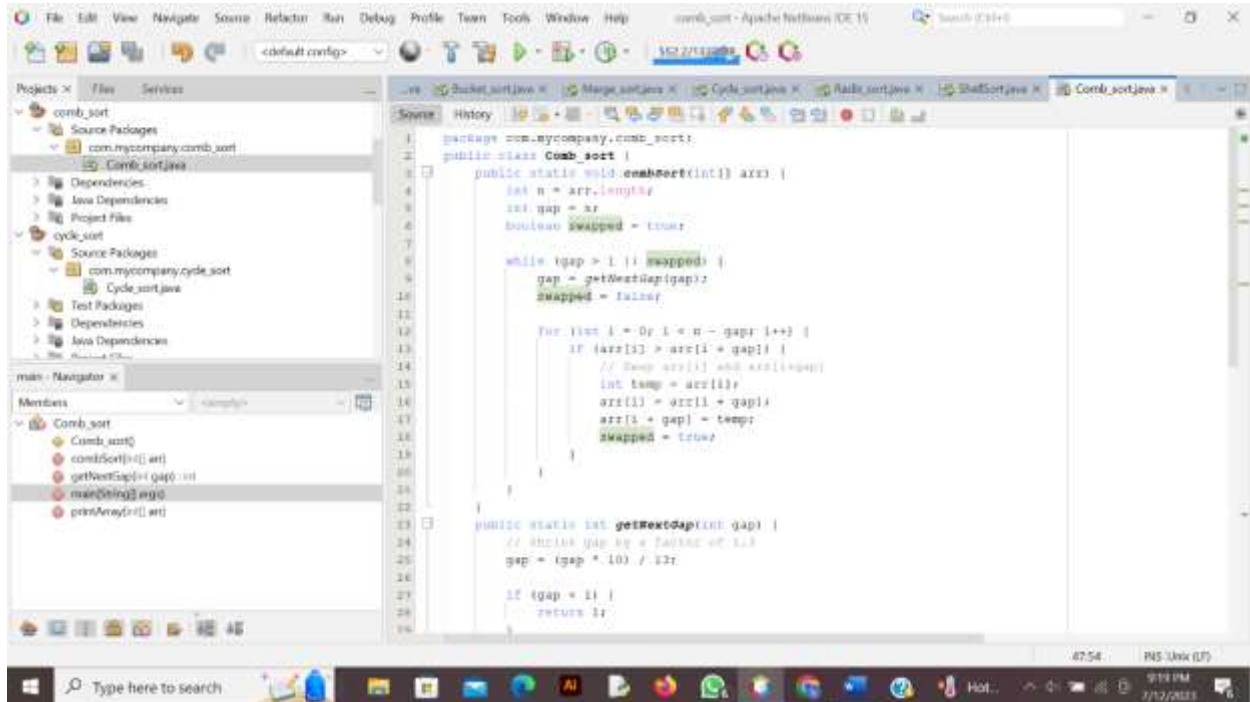
The screenshot shows the output of the Shell Sort program in the Output window:

```
-----< com.mycompany:shell-sort >-----
Building shell-sort 1.0-SNAPSHOT
-----[ jar ]-----
--- exec-maven-plugin:2.5.1:exec (default-cli) @ shell-sort ---
Before Sorted Array :
[33, 22, 88, 55, 44, 99, 77, 66, 11, 10]

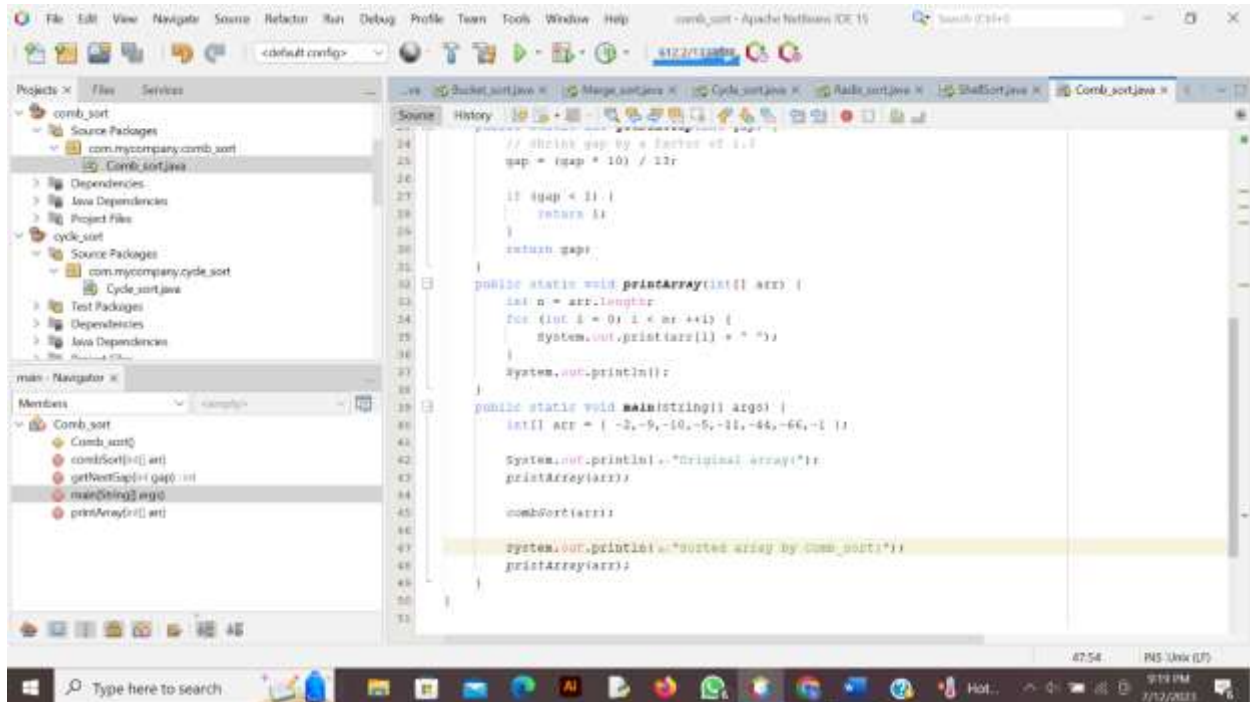
After Sorting Array by Shell Sort :
[10, 11, 22, 33, 44, 55, 66, 77, 88, 99]

BUILD SUCCESS
Total time: 0.863 s
Finished at: 2023-07-12T20:54:07-07:00
-----
```

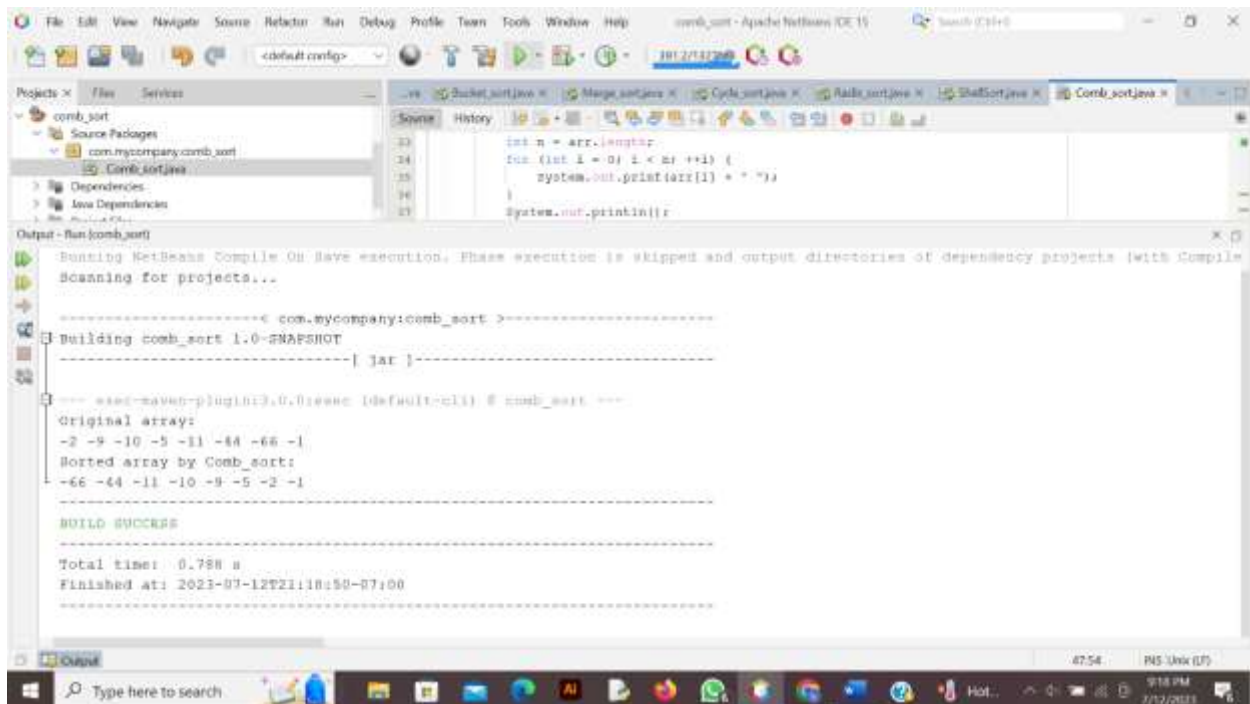
# Shell Sort



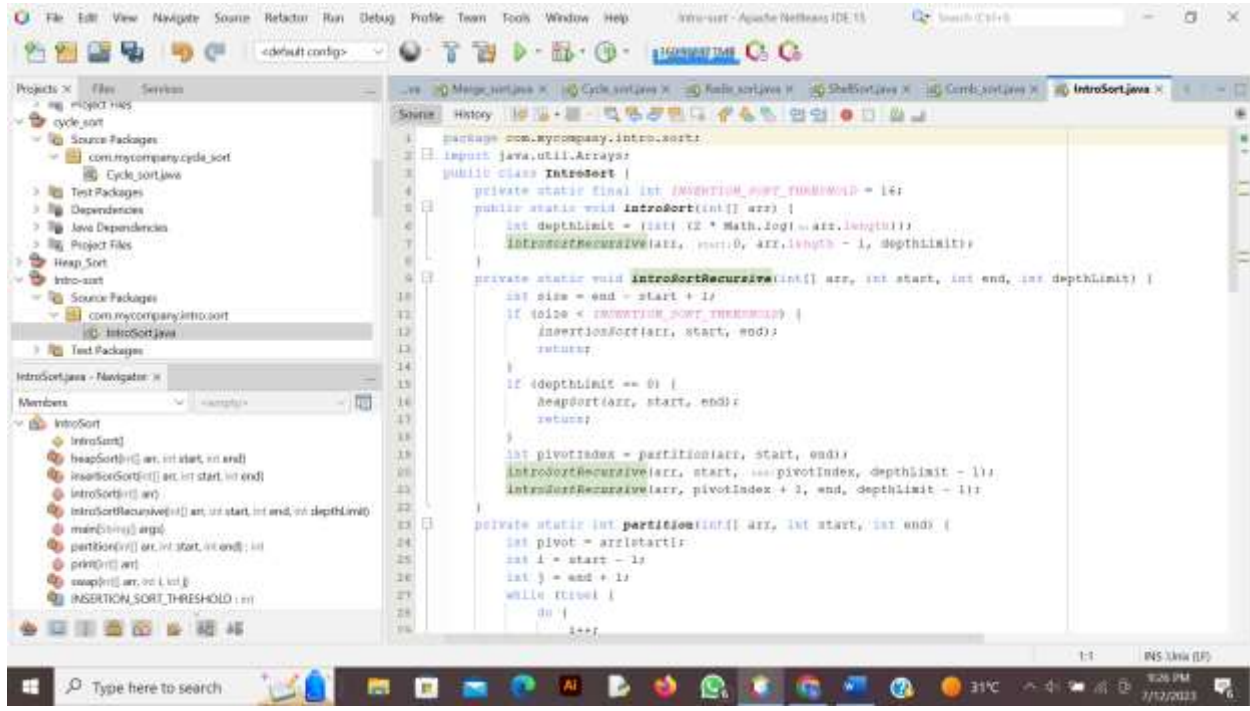
```
1 package com.mycompany.comb_sort;
2 public class Comb_sort {
3     public static void combSort(int[] arr) {
4         int n = arr.length;
5         int gap = n;
6         boolean swapped = true;
7
8         while (gap > 1 || swapped) {
9             gap = getNextGap(gap);
10            swapped = false;
11
12            for (int i = 0; i < n - gap; i++) {
13                if (arr[i] > arr[i + gap]) {
14                    // Swap arr[i] and arr[i+gap]
15                    int temp = arr[i];
16                    arr[i] = arr[i + gap];
17                    arr[i + gap] = temp;
18                    swapped = true;
19                }
20            }
21        }
22
23        public static int getNextGap(int gap) {
24            // Shrink gap by a factor of 1.3
25            gap = (gap * 10) / 13;
26
27            if (gap < 1) {
28                return 1;
29            }
30            return gap;
31        }
32
33        public static void printArray(int[] arr) {
34            int n = arr.length;
35            for (int i = 0; i < n; i++) {
36                System.out.print(arr[i] + " ");
37            }
38            System.out.println();
39        }
40
41        public static void main(String[] args) {
42            int[] arr = { -2, -9, -10, -5, -11, -46, -66, -1 };
43
44            System.out.println("Original array:");
45            printArray(arr);
46
47            combSort(arr);
48
49            System.out.println("Sorted array by Comb_Sort:");
50            printArray(arr);
51        }
52    }
53 }
```

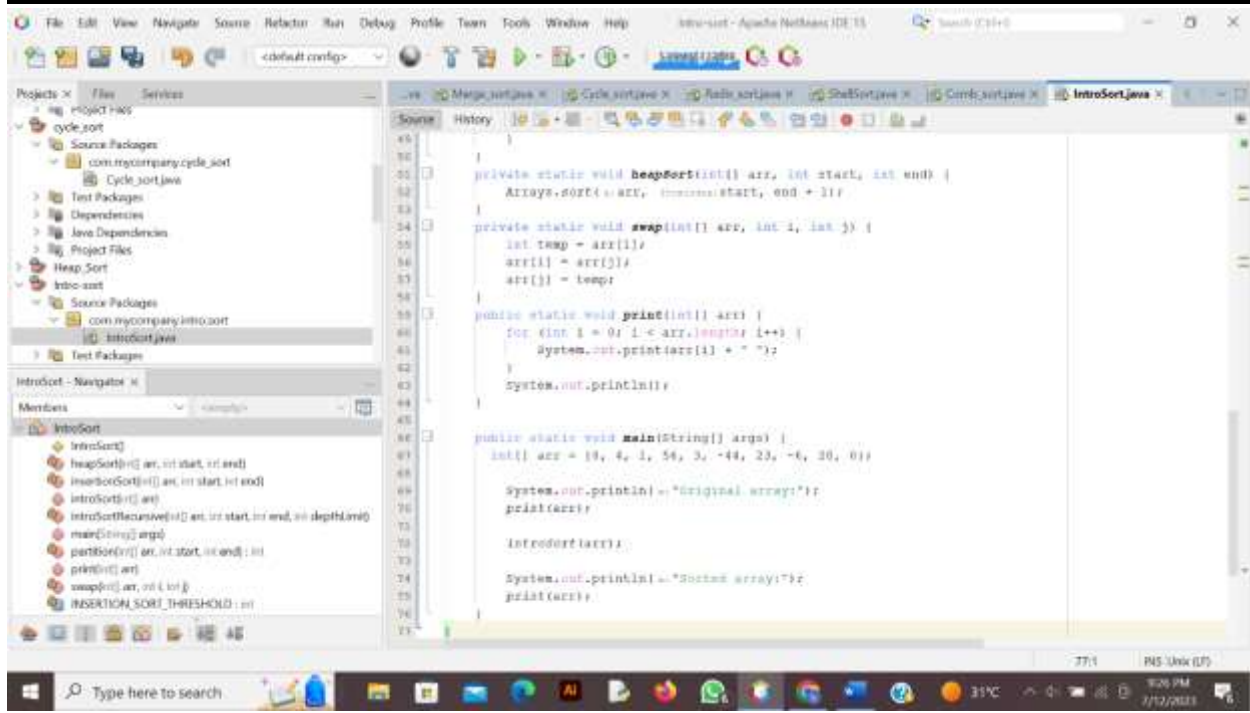
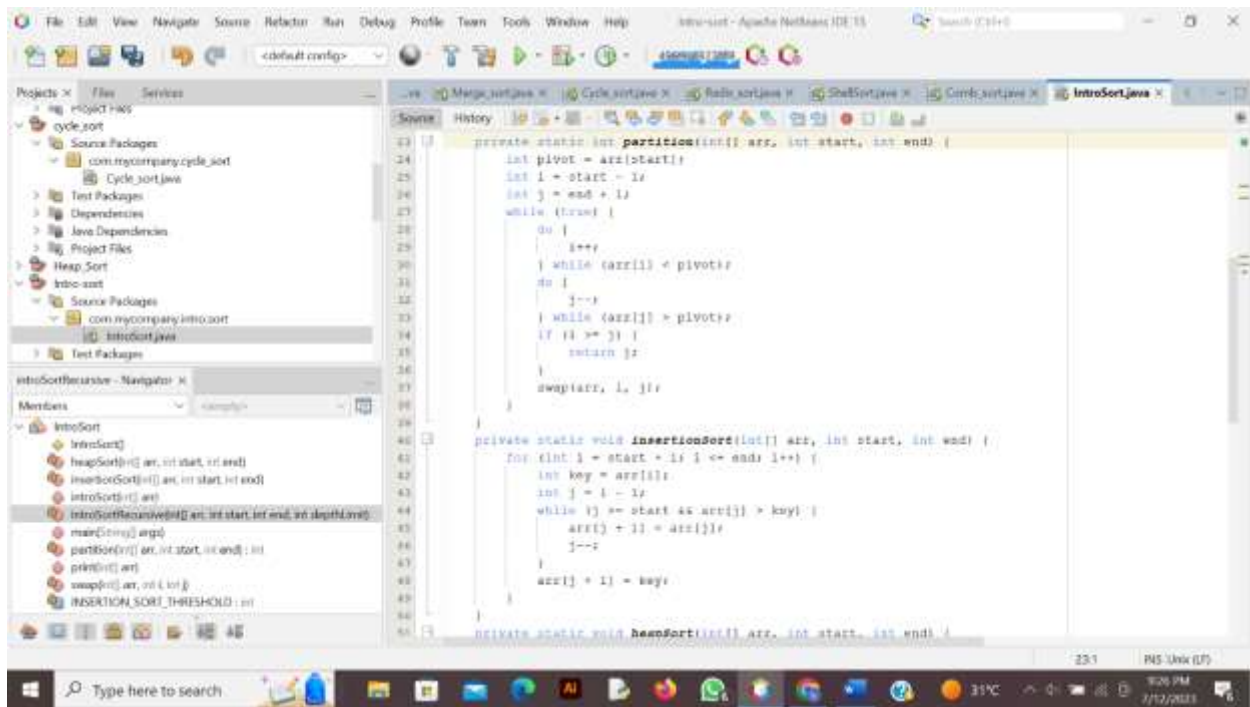


```
24 // Shrink gap by a factor of 1.3
25 gap = (gap * 10) / 13;
26
27 if (gap < 1) {
28     return 1;
29 }
30 return gap;
31 }
32
33 public static void printArray(int[] arr) {
34     int n = arr.length;
35     for (int i = 0; i < n; i++) {
36         System.out.print(arr[i] + " ");
37     }
38     System.out.println();
39 }
40
41 public static void main(String[] args) {
42     int[] arr = { -2, -9, -10, -5, -11, -46, -66, -1 };
43
44     System.out.println("Original array:");
45     printArray(arr);
46
47     combSort(arr);
48
49     System.out.println("Sorted array by Comb_Sort:");
50     printArray(arr);
51 }
52 }
```

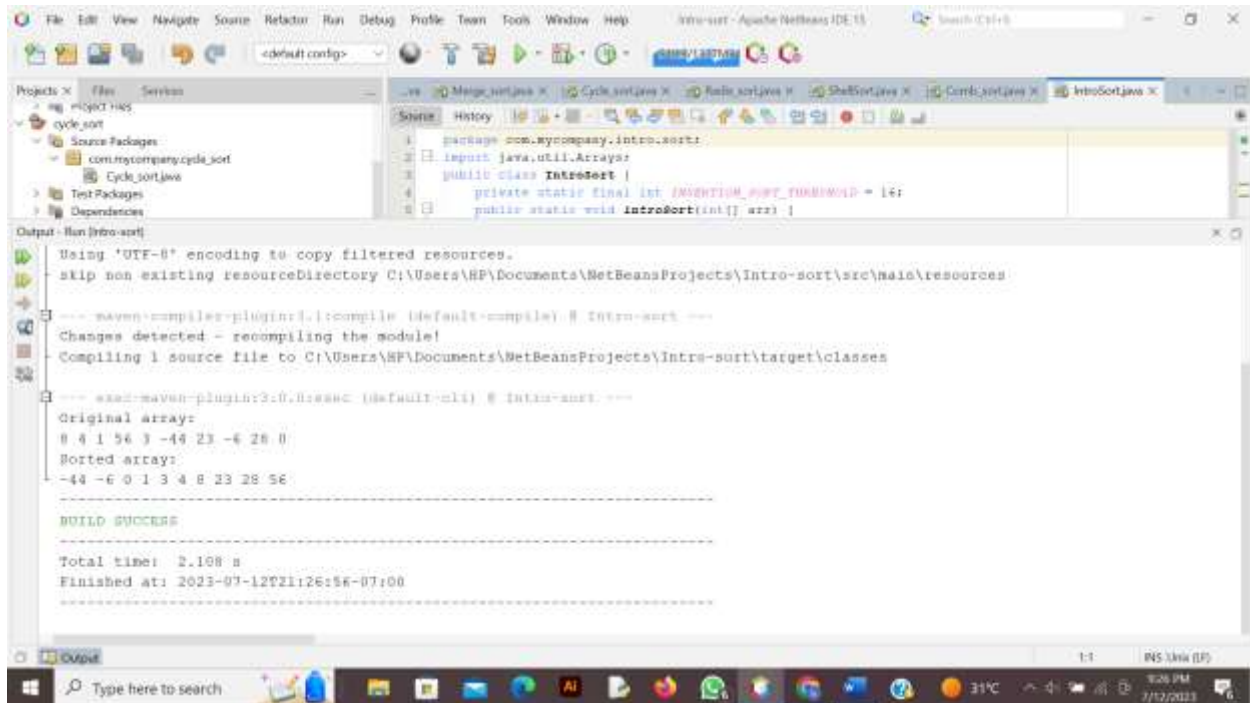


## Intro Sort

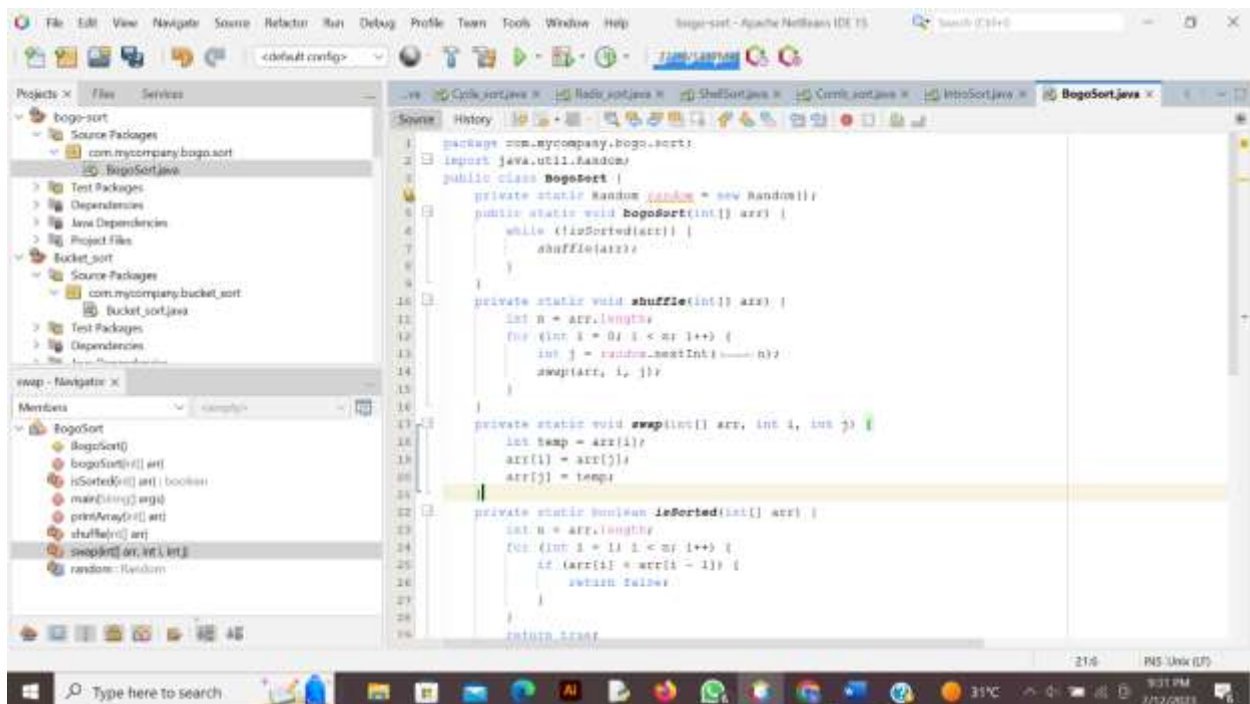


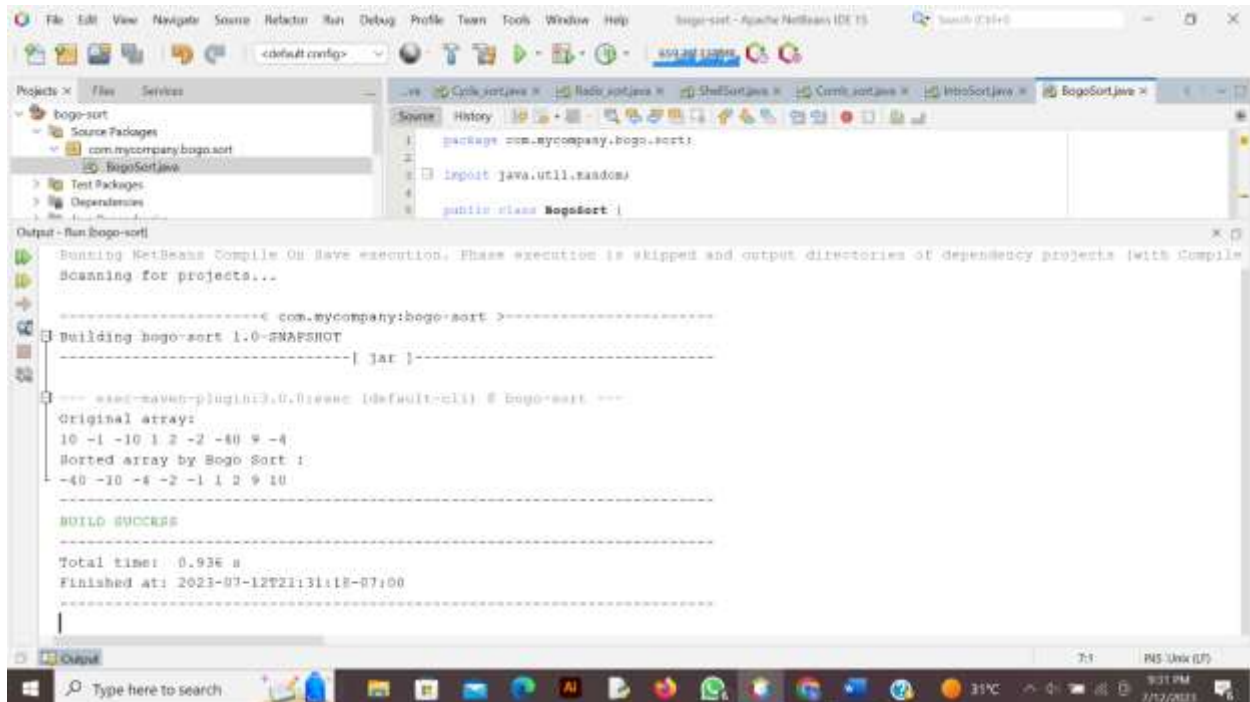
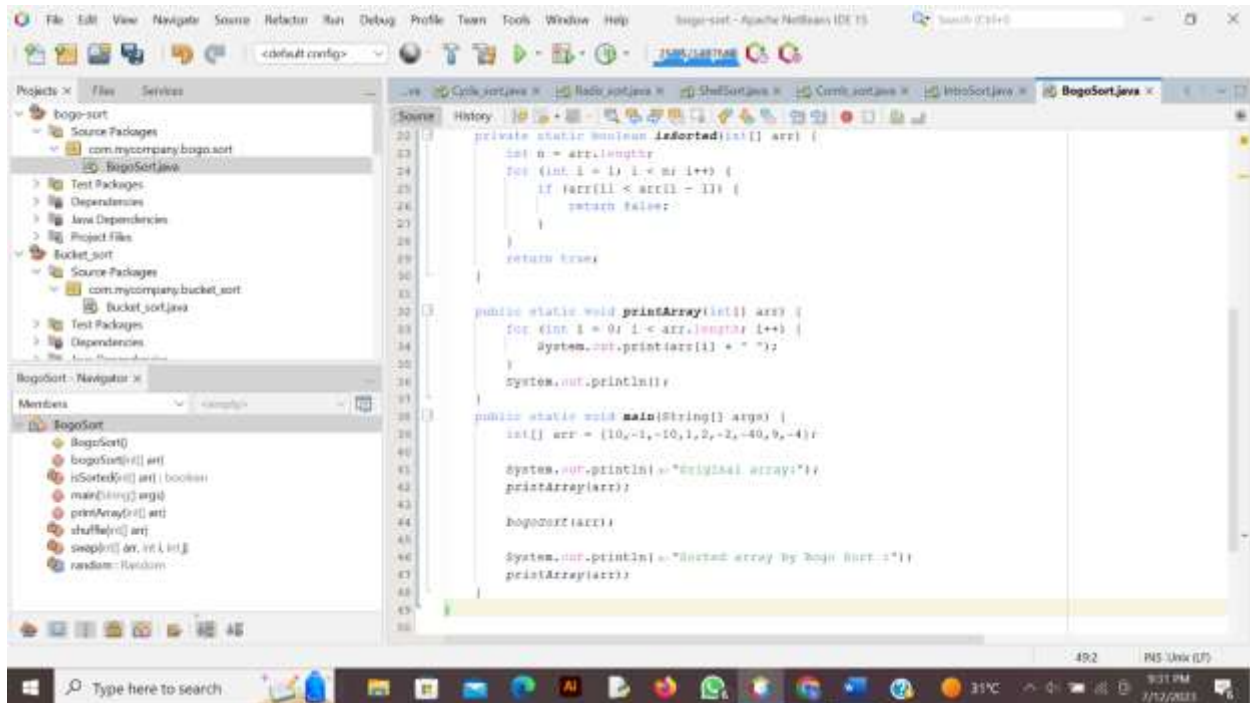




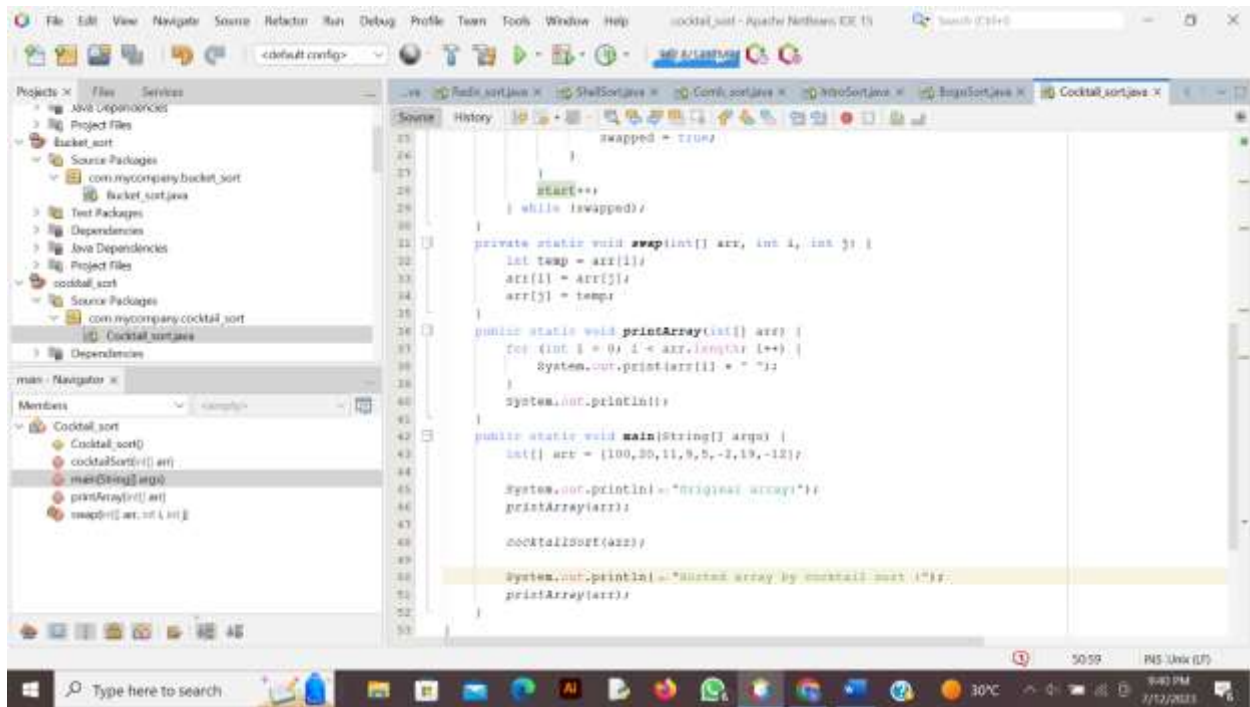
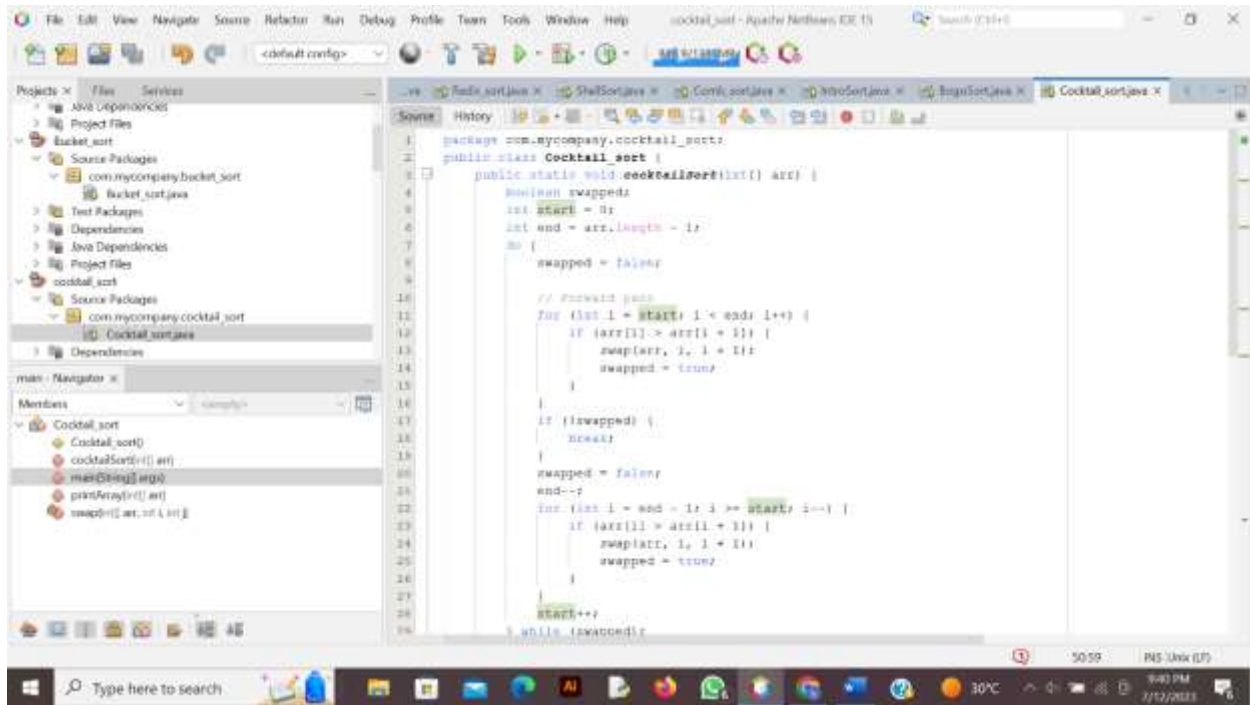


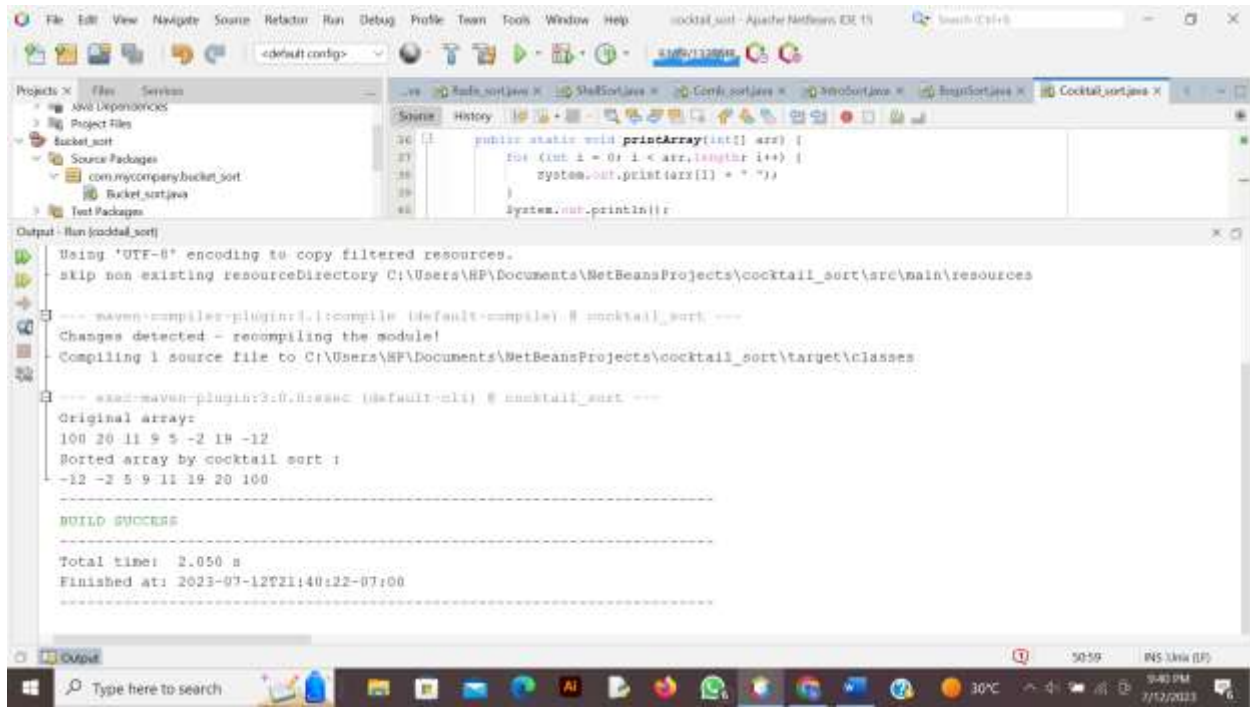
## Bogo Sort



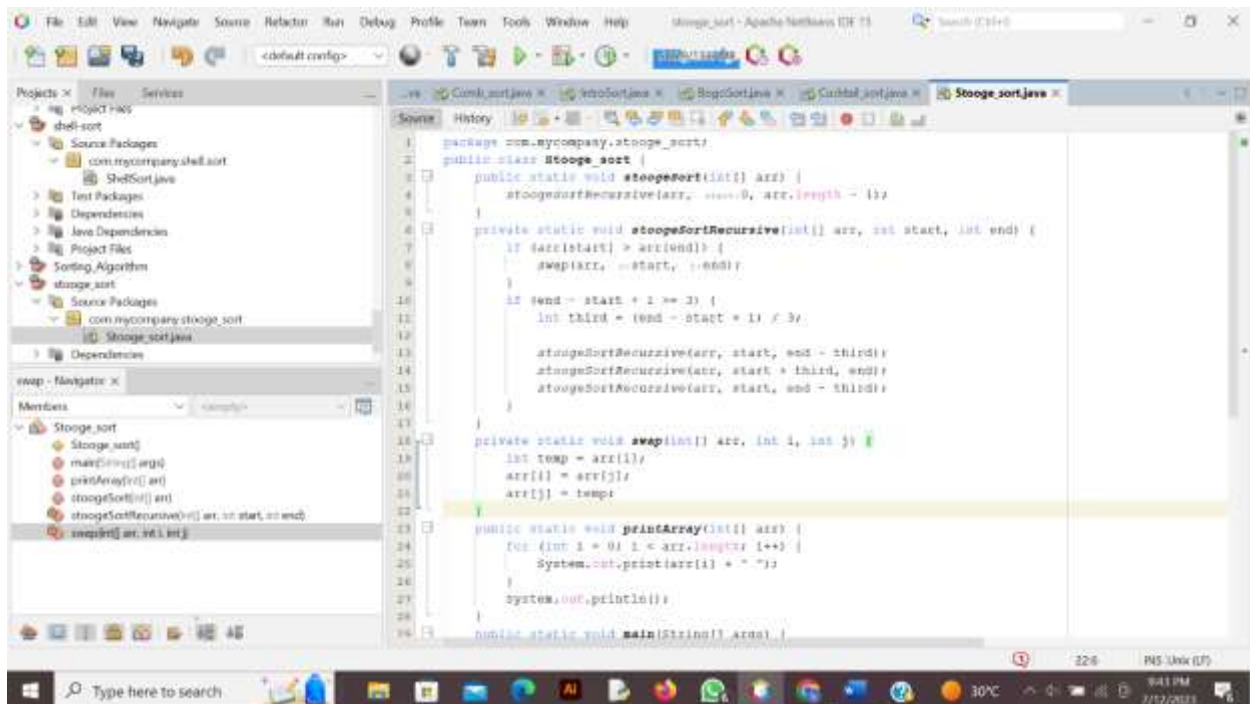


## Cocktail Sort

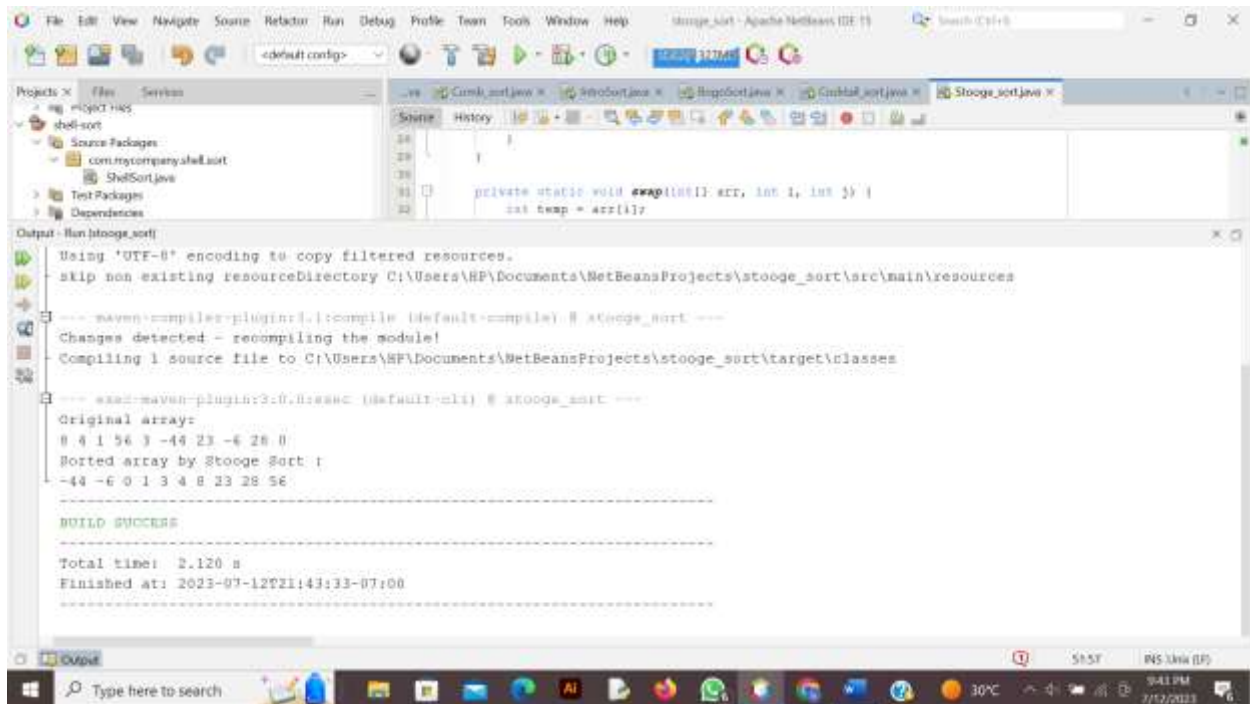
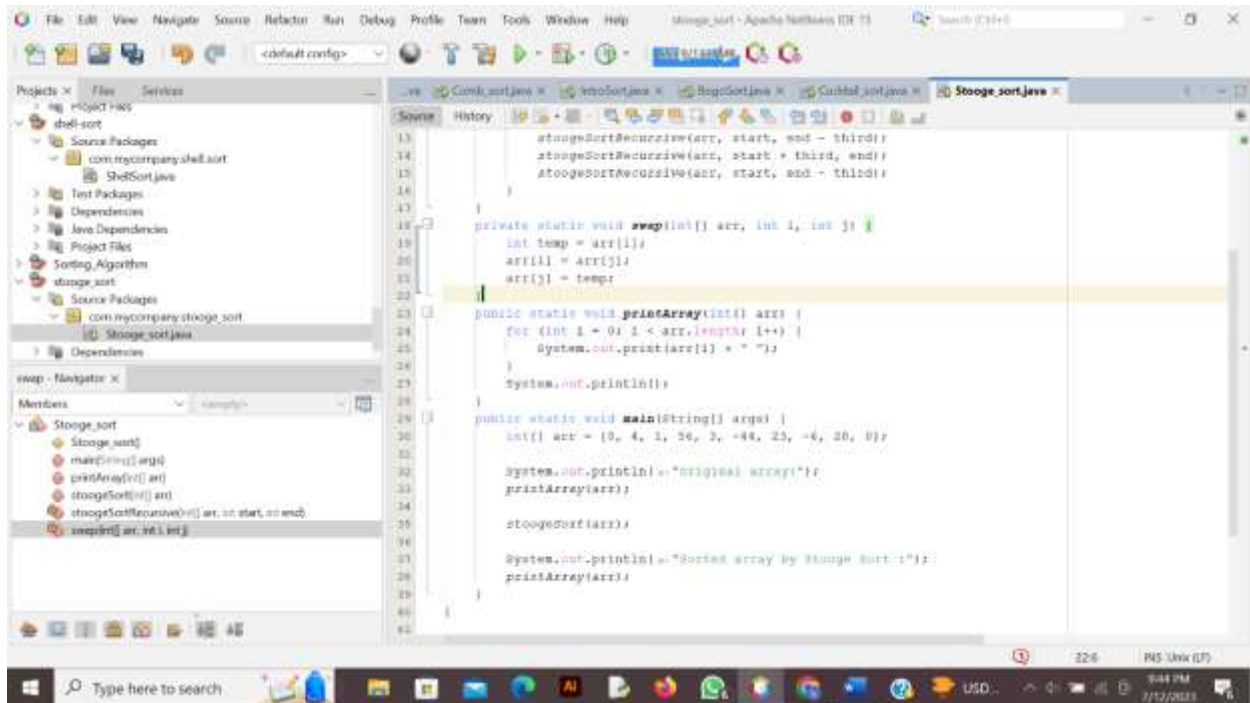




## Stooge Sort







## Pigeonhole Sort

