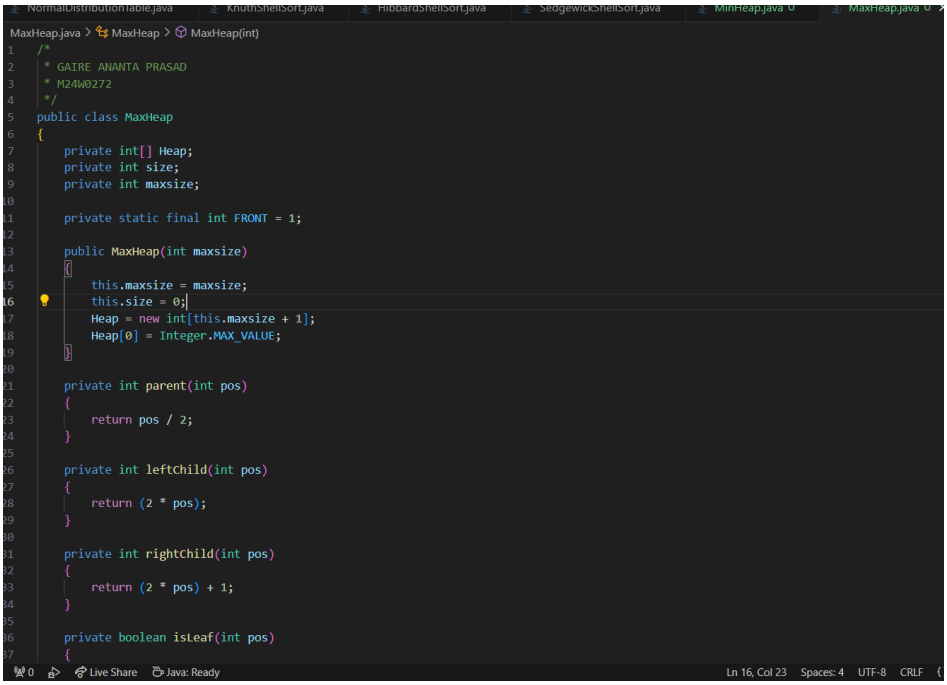
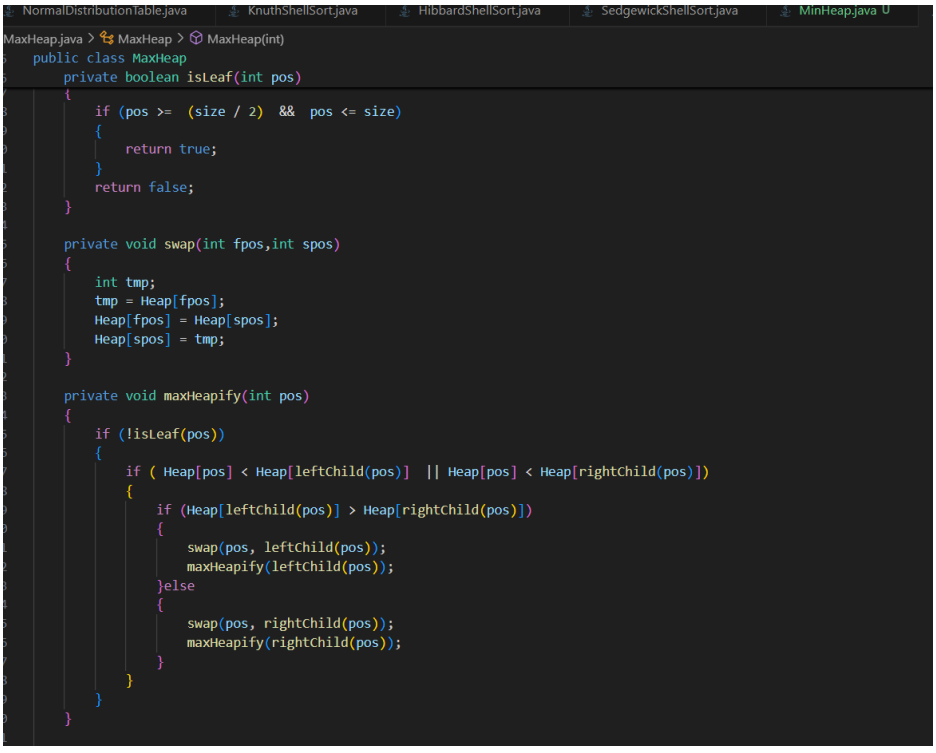
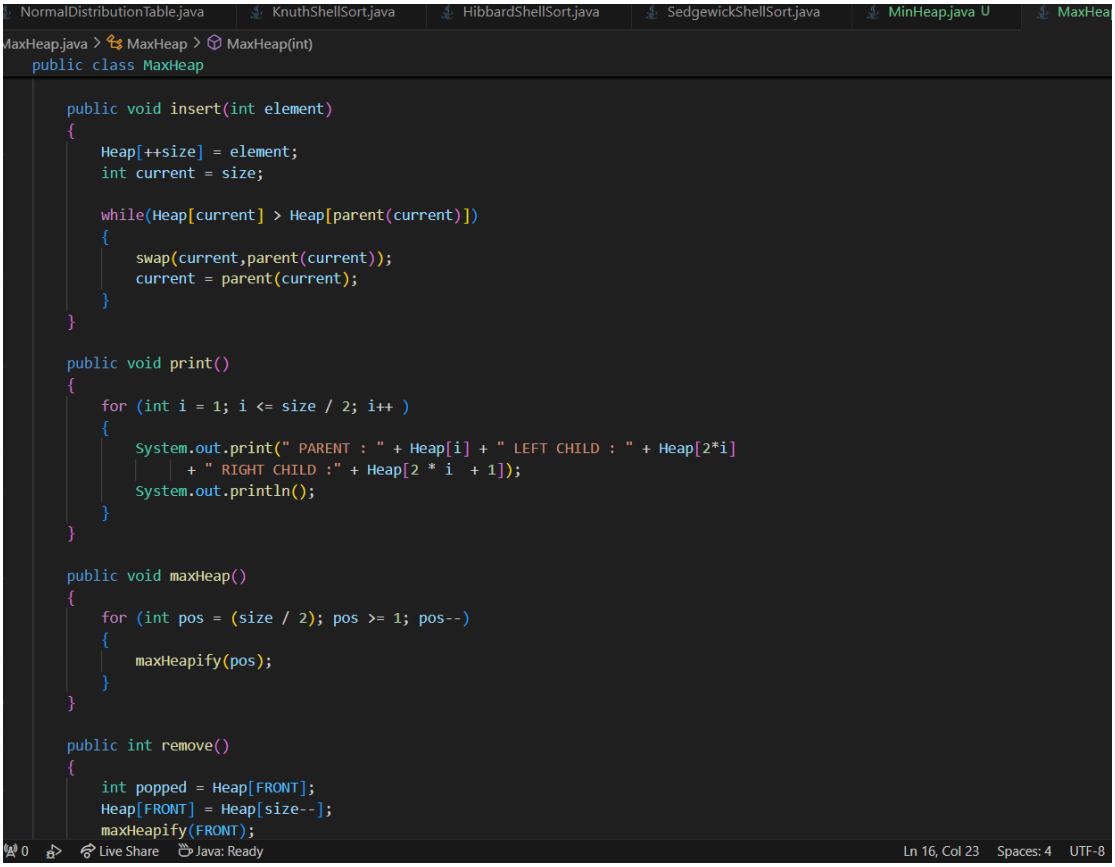
GAIRE ANANTA PRASAD
M24W0272
CODE FOR MAX-HEAP

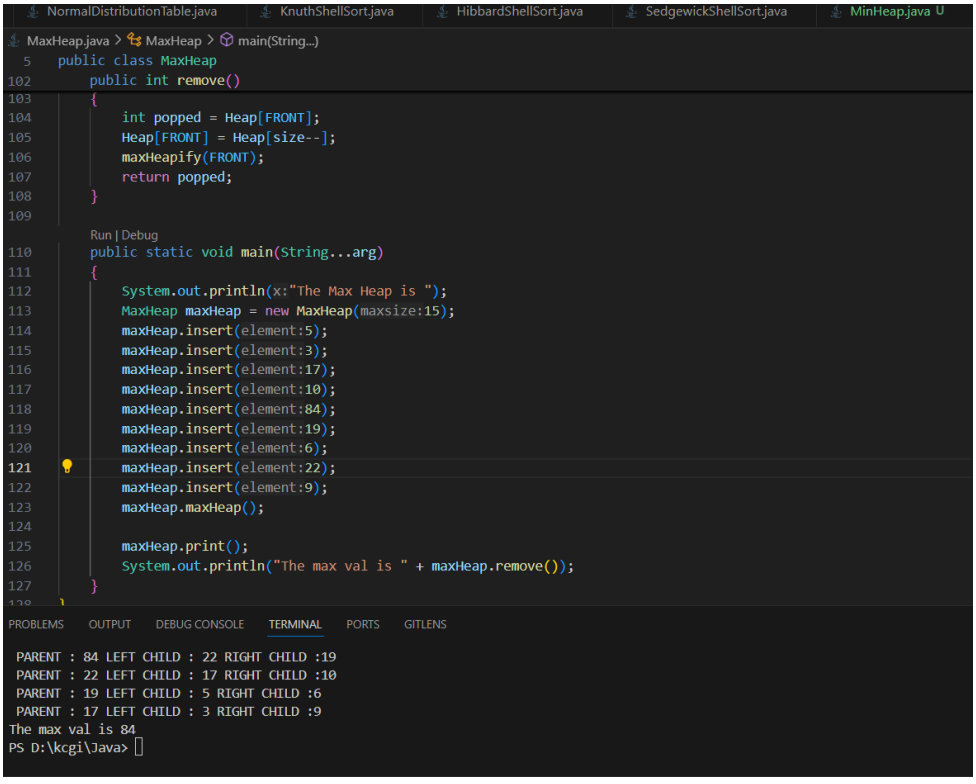


```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */
public class MaxHeap
{
    private int[] Heap;
    private int size;
    private int maxsize;

    private static final int FRONT = 1;

    public MaxHeap(int maxsize)
    {
        this.maxsize = maxsize;
        this.size = 0;
        Heap = new int[this.maxsize + 1];
        Heap[0] = Integer.MAX_VALUE;
    }

    private int parent(int pos)
    {
        return pos / 2;
    }

    private int leftChild(int pos)
    {
        return (2 * pos);
    }

    private int rightChild(int pos)
    {
        return (2 * pos) + 1;
    }

    private boolean isLeaf(int pos)
    {
```



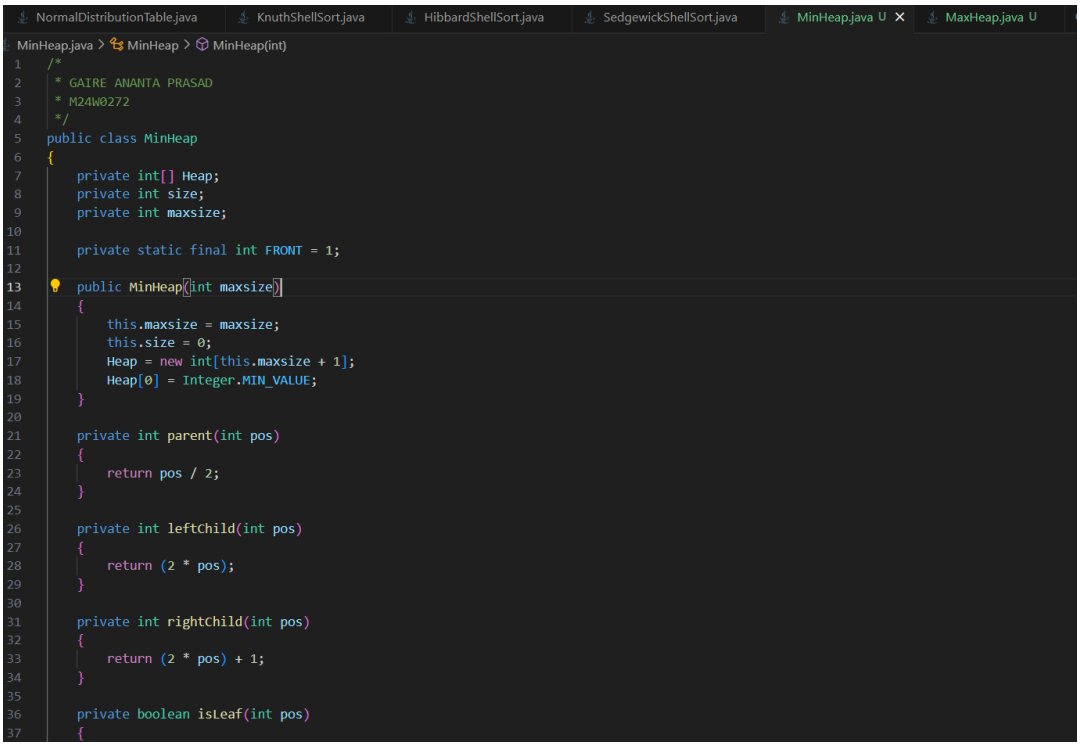
```java
public class MaxHeap
    private boolean isLeaf(int pos)
    {
        if (pos >=  (size / 2)  &&  pos <= size)
        {
            return true;
        }
        return false;
    }

    private void swap(int fpos,int spos)
    {
        int tmp;
        tmp = Heap[fpos];
        Heap[fpos] = Heap[spos];
        Heap[spos] = tmp;
    }

    private void maxHeapify(int pos)
    {
        if (!isLeaf(pos))
        {
            if ( Heap[pos] < Heap[leftChild(pos)]  || Heap[pos] < Heap[rightChild(pos)])
            {
                if (Heap[leftChild(pos)] > Heap[rightChild(pos)])
                {
                    swap(pos, leftChild(pos));
                    maxHeapify(leftChild(pos));
                }else
                {
                    swap(pos, rightChild(pos));
                    maxHeapify(rightChild(pos));
                }
            }
        }
    }
```



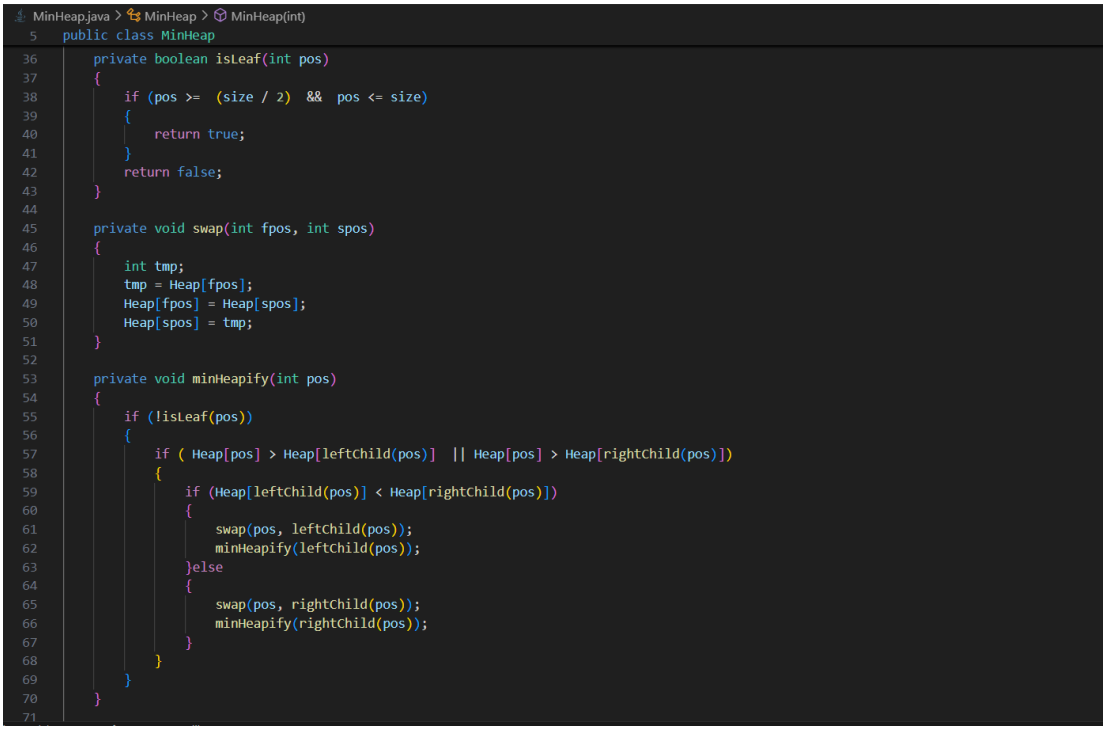


Gaire Ananta Prasad
M24W0272

CODE FOR MIN-HEAP

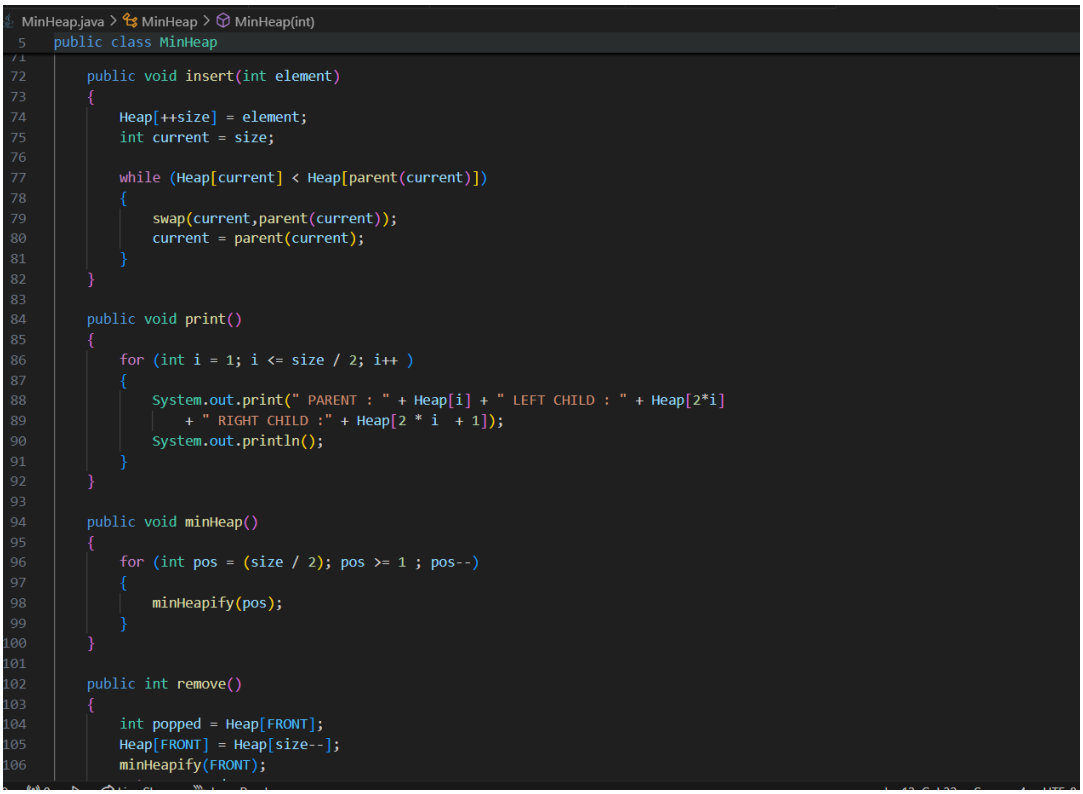


```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */
public class MinHeap
{
    private int[] Heap;
    private int size;
    private int maxsize;

    private static final int FRONT = 1;

    public MinHeap(int maxsize)
    {
        this.maxsize = maxsize;
        this.size = 0;
        Heap = new int[this.maxsize + 1];
        Heap[0] = Integer.MIN_VALUE;
    }

    private int parent(int pos)
    {
        return pos / 2;
    }

    private int leftChild(int pos)
    {
        return (2 * pos);
    }

    private int rightChild(int pos)
    {
        return (2 * pos) + 1;
    }

    private boolean isLeaf(int pos)
    {
```

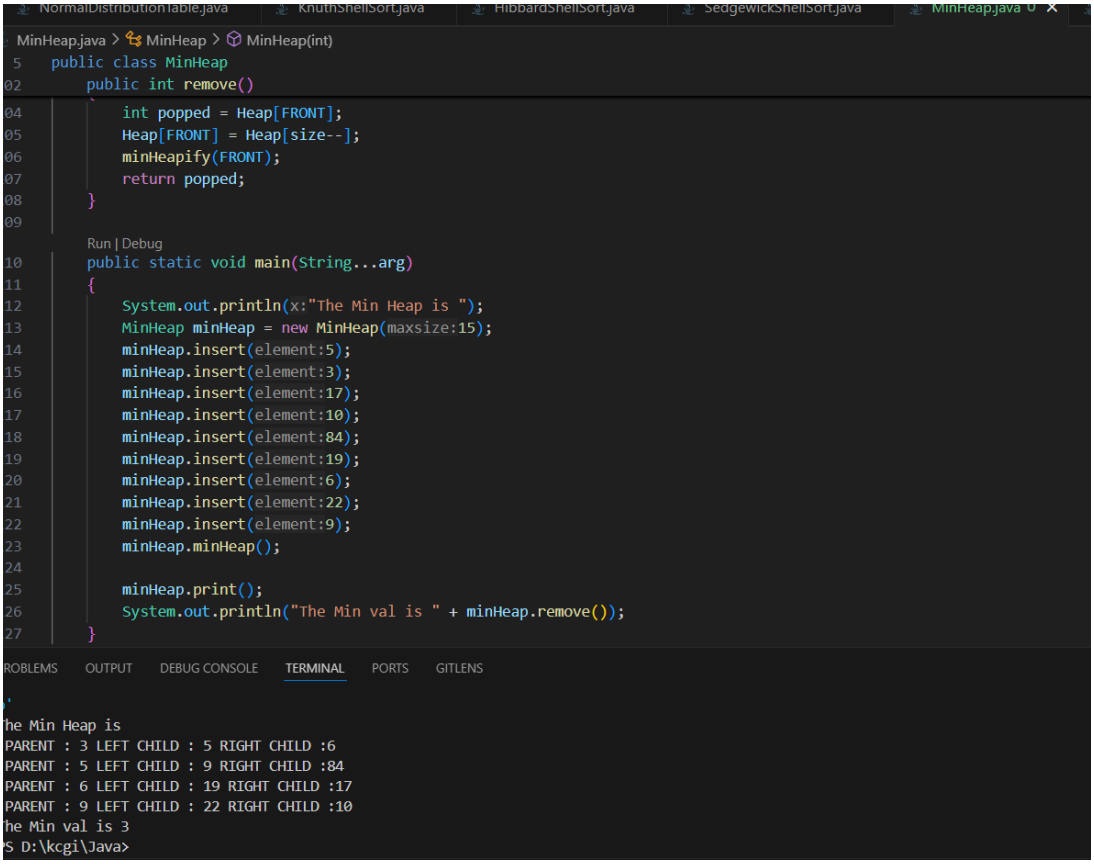


```java
    private boolean isLeaf(int pos)
    {
        if (pos >=  (size / 2)  &&  pos <= size)
        {
            return true;
        }
        return false;
    }

    private void swap(int fpos, int spos)
    {
        int tmp;
        tmp = Heap[fpos];
        Heap[fpos] = Heap[spos];
        Heap[spos] = tmp;
    }

    private void minHeapify(int pos)
    {
        if (!isLeaf(pos))
        {
            if ( Heap[pos] > Heap[leftChild(pos)]  || Heap[pos] > Heap[rightChild(pos)])
            {
                if (Heap[leftChild(pos)] < Heap[rightChild(pos)])
                {
                    swap(pos, leftChild(pos));
                    minHeapify(leftChild(pos));
                }else
                {
                    swap(pos, rightChild(pos));
                    minHeapify(rightChild(pos));
                }
            }
        }
    }
```

Gaire Ananta Prasad
M24W0272



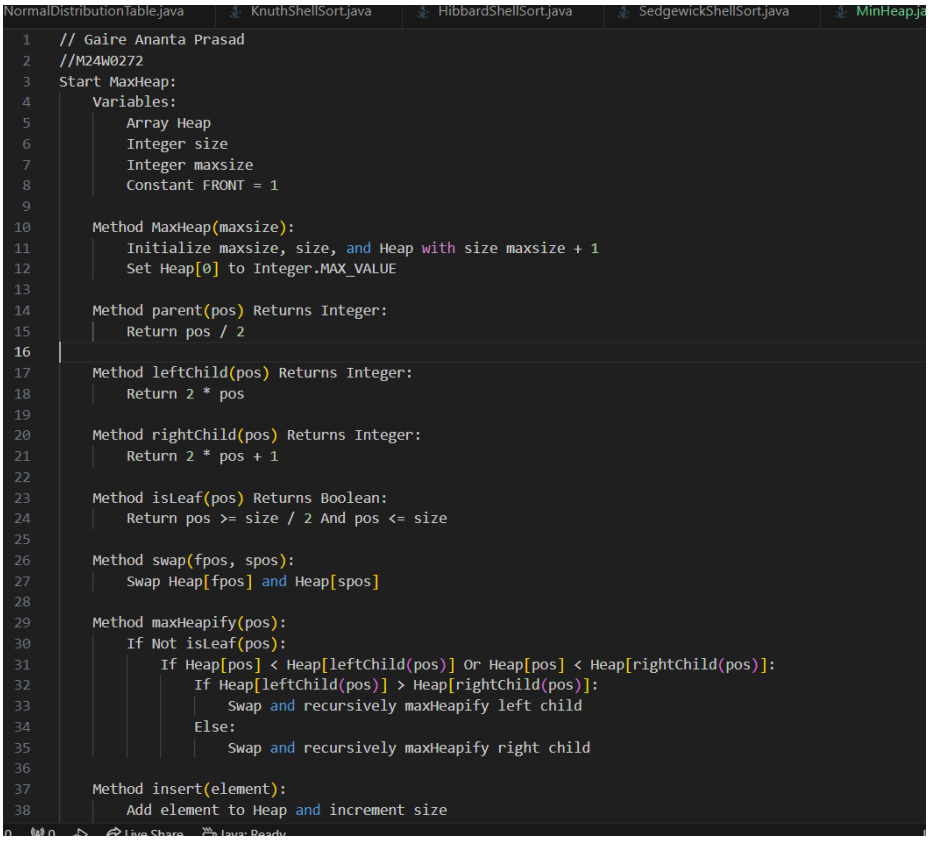
```
MinHeap.java > MinHeap > MinHeap(int)
public class MinHeap

    public void insert(int element)
    {
        Heap[++size] = element;
        int current = size;

        while (Heap[current] < Heap[parent(current)])
        {
            swap(current,parent(current));
            current = parent(current);
        }
    }

    public void print()
    {
        for (int i = 1; i <= size / 2; i++ )
        {
            System.out.print(" PARENT : " + Heap[i] + " LEFT CHILD : " + Heap[2*i]
                + " RIGHT CHILD :" + Heap[2 * i  + 1]);
            System.out.println();
        }
    }

    public void minHeap()
    {
        for (int pos = (size / 2); pos >= 1 ; pos--)
        {
            minHeapify(pos);
        }
    }

    public int remove()
    {
        int popped = Heap[FRONT];
        Heap[FRONT] = Heap[size--];
        minHeapify(FRONT);
```



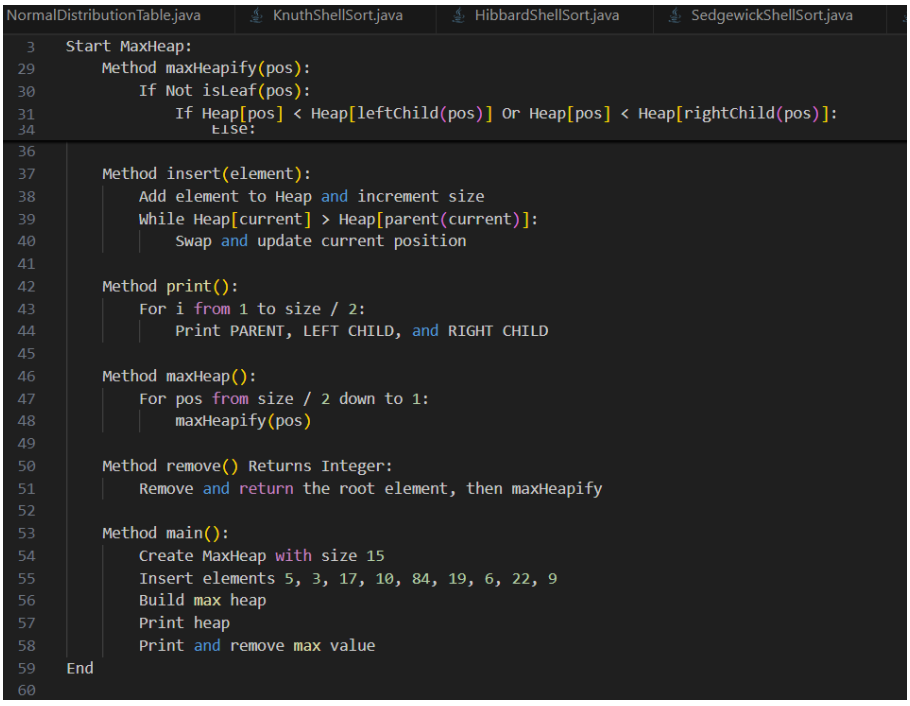
```
NormalDistributionTable.java   KnuthShellSort.java   HibbardShellSort.java   SedgewickShellSort.java   MinHeap.java U
MinHeap.java > MinHeap > MinHeap(int)
public class MinHeap
    public int remove()
        int popped = Heap[FRONT];
        Heap[FRONT] = Heap[size--];
        minHeapify(FRONT);
        return popped;
    }

    Run | Debug
    public static void main(String...arg)
    {
        System.out.println(x:"The Min Heap is ");
        MinHeap minHeap = new MinHeap(maxsize:15);
        minHeap.insert(element:5);
        minHeap.insert(element:3);
        minHeap.insert(element:17);
        minHeap.insert(element:10);
        minHeap.insert(element:84);
        minHeap.insert(element:19);
        minHeap.insert(element:6);
        minHeap.insert(element:22);
        minHeap.insert(element:9);
        minHeap.minHeap();

        minHeap.print();
        System.out.println("The Min val is " + minHeap.remove());
    }
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS
'
he Min Heap is
PARENT : 3 LEFT CHILD : 5 RIGHT CHILD :6
PARENT : 5 LEFT CHILD : 9 RIGHT CHILD :84
PARENT : 6 LEFT CHILD : 19 RIGHT CHILD :17
PARENT : 9 LEFT CHILD : 22 RIGHT CHILD :10
he Min val is 3
S D:\kcgi\Java>
```

Gaire Ananta Prasad
M24W0272

Pseudocode for MaxHeap



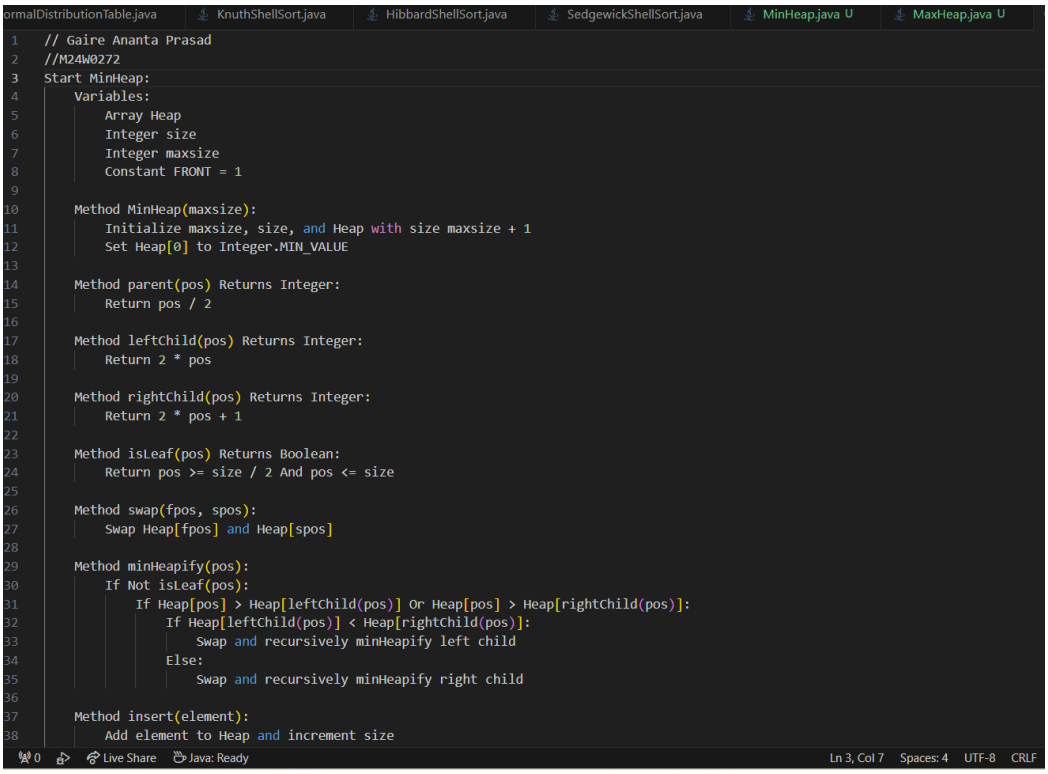
```
// Gaire Ananta Prasad
//M24W0272
Start MaxHeap:
    Variables:
        Array Heap
        Integer size
        Integer maxsize
        Constant FRONT = 1

    Method MaxHeap(maxsize):
        Initialize maxsize, size, and Heap with size maxsize + 1
        Set Heap[0] to Integer.MAX_VALUE

    Method parent(pos) Returns Integer:
        Return pos / 2

    Method leftChild(pos) Returns Integer:
        Return 2 * pos

    Method rightChild(pos) Returns Integer:
        Return 2 * pos + 1

    Method isLeaf(pos) Returns Boolean:
        Return pos >= size / 2 And pos <= size

    Method swap(fpos, spos):
        Swap Heap[fpos] and Heap[spos]

    Method maxHeapify(pos):
        If Not isLeaf(pos):
            If Heap[pos] < Heap[leftChild(pos)] Or Heap[pos] < Heap[rightChild(pos)]:
                If Heap[leftChild(pos)] > Heap[rightChild(pos)]:
                    Swap and recursively maxHeapify left child
                Else:
                    Swap and recursively maxHeapify right child

    Method insert(element):
        Add element to Heap and increment size
```

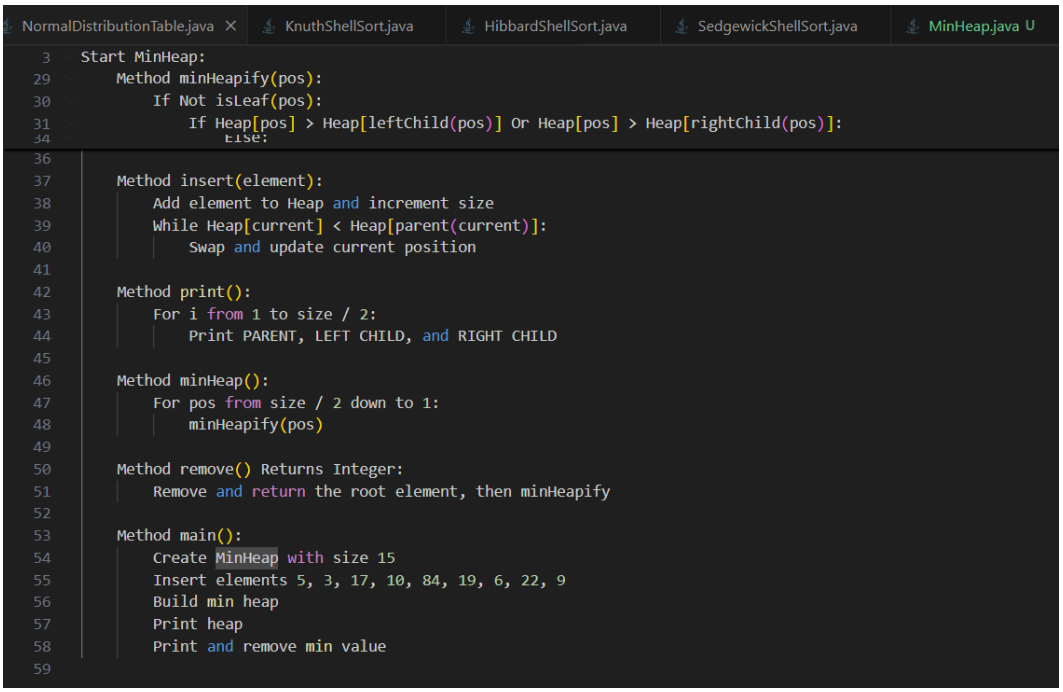


```
Start MaxHeap:
    Method maxHeapify(pos):
        If Not isLeaf(pos):
            If Heap[pos] < Heap[leftChild(pos)] Or Heap[pos] < Heap[rightChild(pos)]:
                Else:

    Method insert(element):
        Add element to Heap and increment size
        While Heap[current] > Heap[parent(current)]:
            Swap and update current position

    Method print():
        For i from 1 to size / 2:
            Print PARENT, LEFT CHILD, and RIGHT CHILD

    Method maxHeap():
        For pos from size / 2 down to 1:
            maxHeapify(pos)

    Method remove() Returns Integer:
        Remove and return the root element, then maxHeapify

    Method main():
        Create MaxHeap with size 15
        Insert elements 5, 3, 17, 10, 84, 19, 6, 22, 9
        Build max heap
        Print heap
        Print and remove max value
End
```

Gaire Ananta Prasad
M24W0272

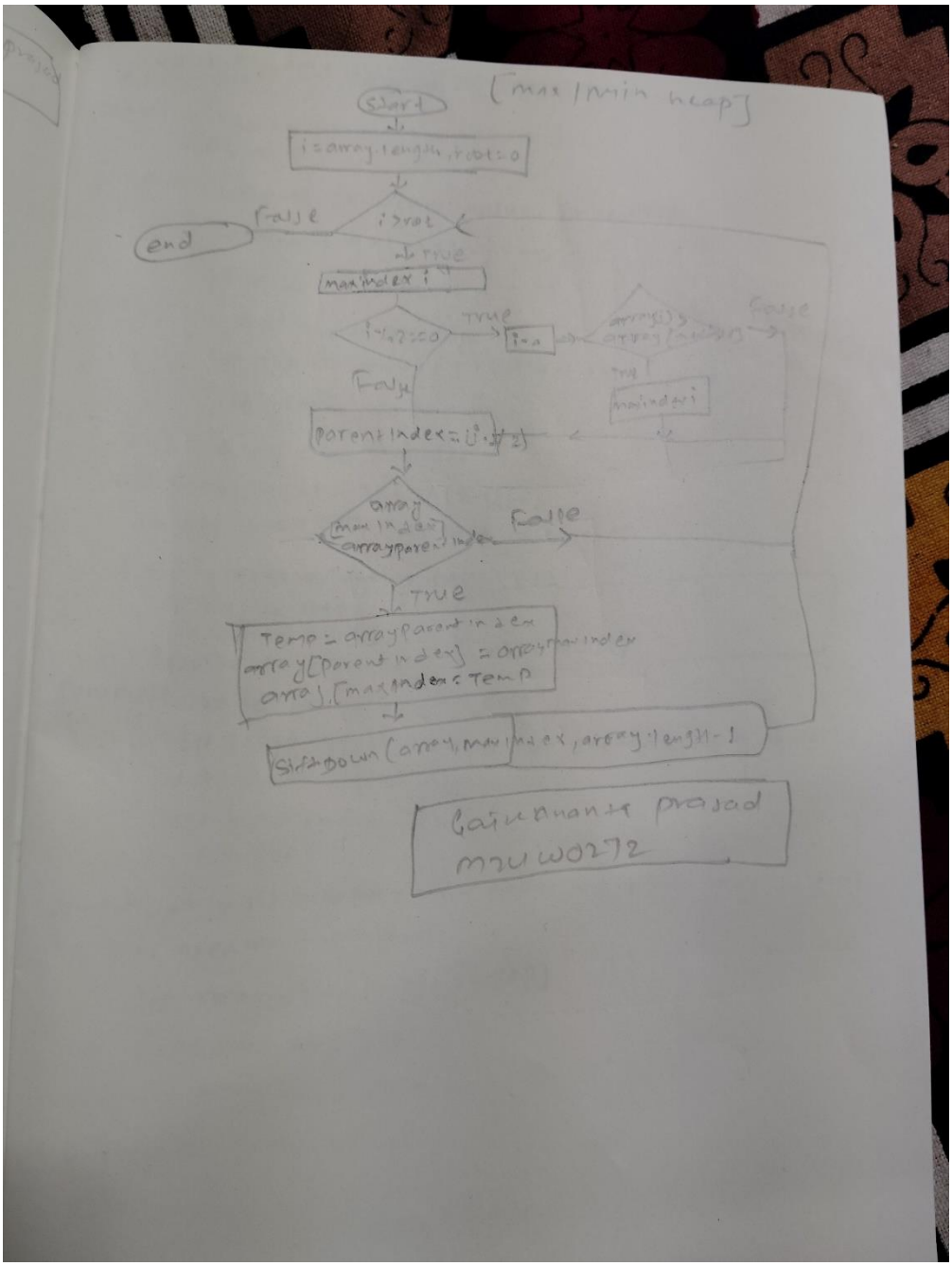
Pseudocode for MaxHeap



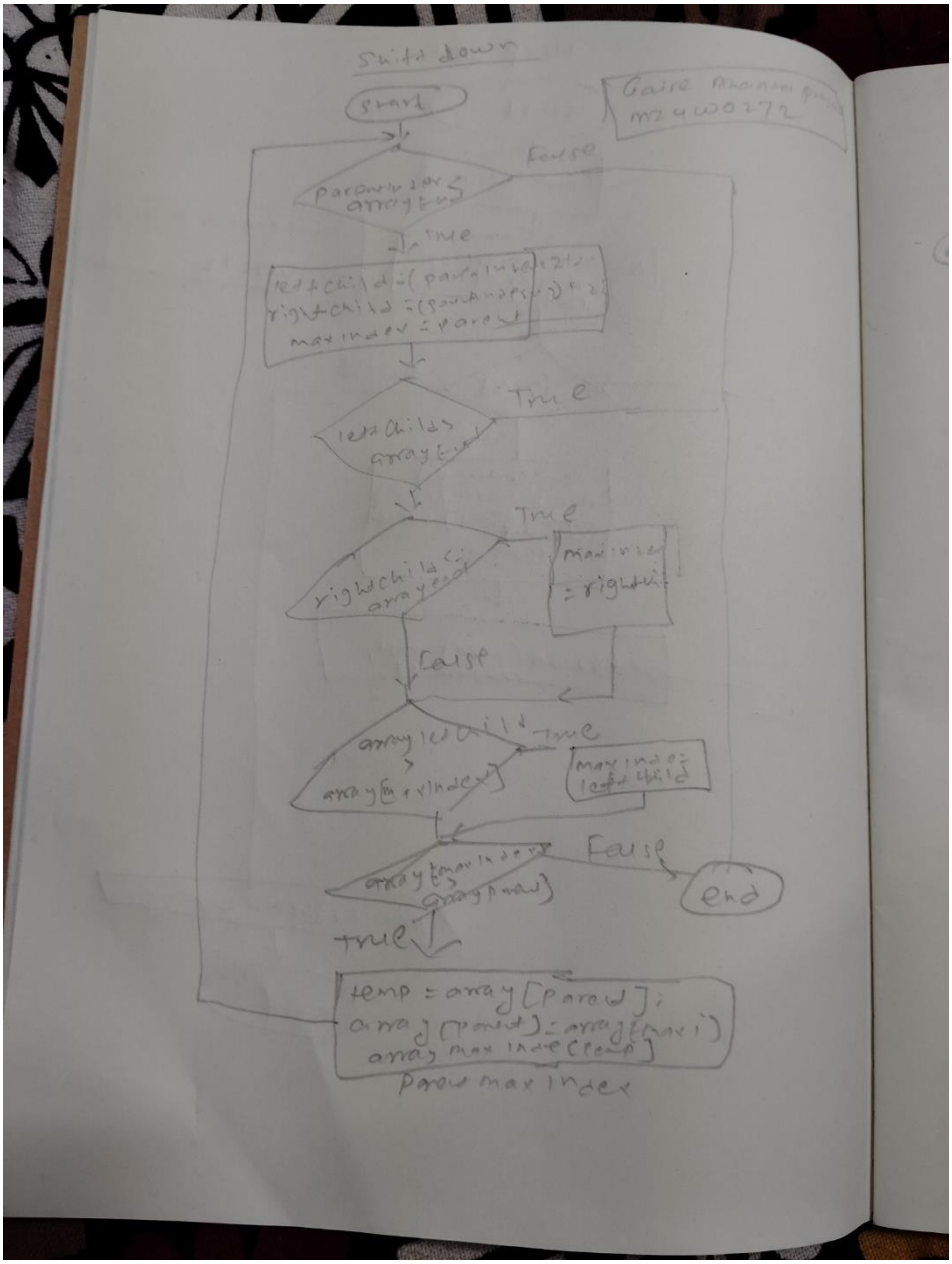


Gaire Ananta Prasad
M24W0272

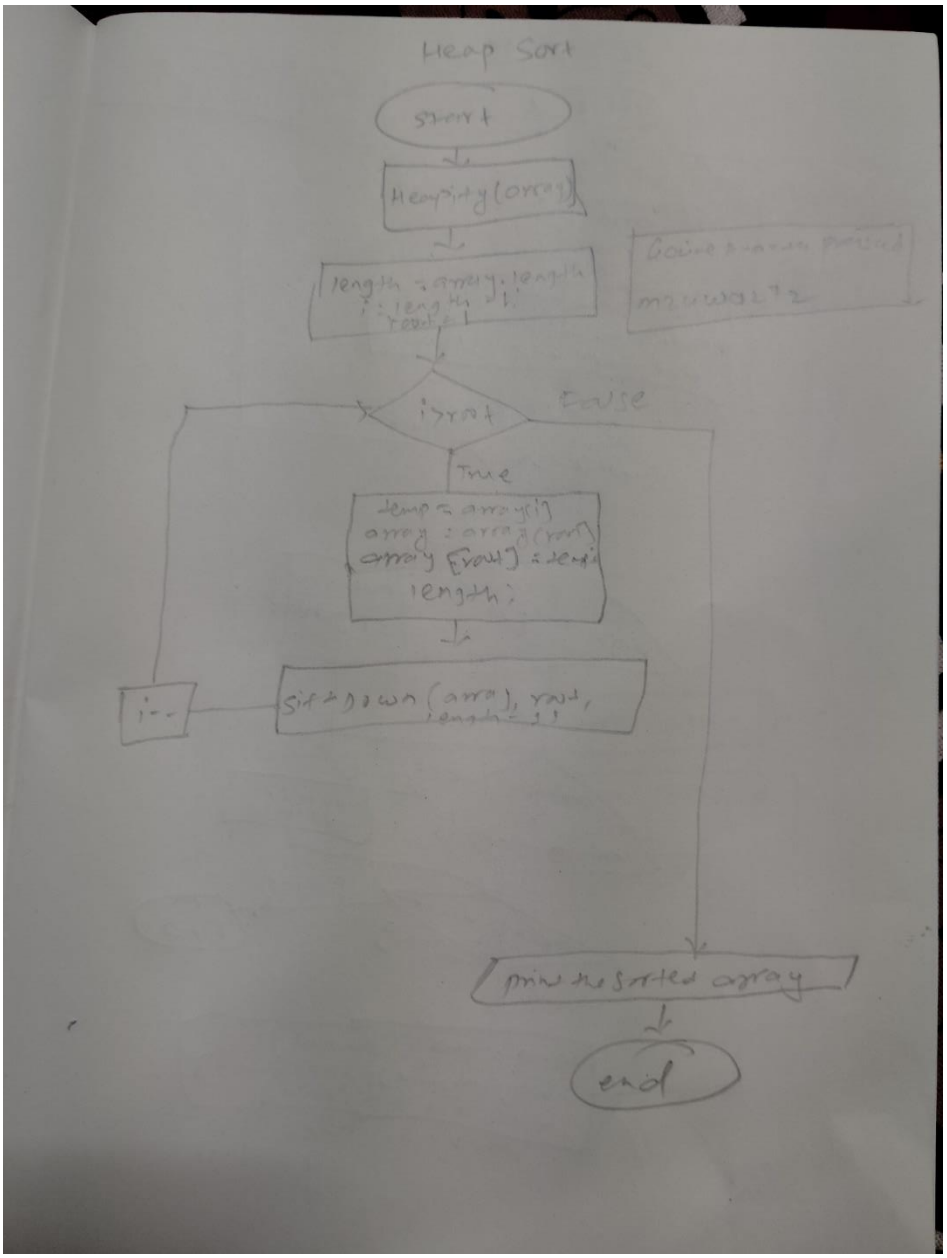
Flowchart for Heap Sort



Gaire Ananta Prasad
M24W0272



Gaire Ananta Prasad
M24W0272



Gaire Ananta Prasad
M24W0272