```java
import java.util.Arrays;
import java.util.Scanner;
//Nhan Vo
//CECS 328- Lab#6
public class main {

   public static void main(String[] args){
        //Part A:
        long RunTime=0;
        long EndTime=0;
        System.out.println("Part A:");
        int min=-100;
        int max=100;
        System.out.println("Please enter a positive integer");
        Scanner scan=new Scanner(System.in);
        String n=scan.nextLine();
        while(!n.matches("\\d+")){
            System.out.println("Your input is not an approriate
integer, Please try again:");
            n = scan.nextLine();
        }
        int[] a=new int[Integer.parseInt(n)];

        main m=new main();
        long heapStart=0;
        long heapEnd=0;
        long SelectionStart=0;
        long SelectionEnd=0;
        long hTimeStore=0;
        long sTimeStore=0;
        for(int i=0;i<100;i++){

            for(int j=0;j<Integer.parseInt(n);j++){
                a[j]=(int)(Math.random()*(max-min+1)+min);
            }
            System.out.println("Generated Array:
"+Arrays.toString(a));
            int[] b=Arrays.copyOfRange(a,0,a.length);

            heapStart=System.nanoTime();
            m.heap_sort(a);
            heapEnd=System.nanoTime();
            hTimeStore=hTimeStore+(heapEnd-heapStart);
```

```
            SelectionStart=System.nanoTime();
            m.selectionSort(b);
            SelectionEnd=System.nanoTime();
            sTimeStore=sTimeStore+(SelectionEnd-SelectionStart);

        }
        System.out.println();
        System.out.println("Average Runtime for Heap sort:
"+hTimeStore/100+" nanoseconds");
        System.out.println("Average Runtime for Selection sort:
"+sTimeStore/100+" nanoseconds");
        //System.out.println("Sorted a: "+Arrays.toString(a)+"\n");

        //Part B:
        System.out.println("Part B:");
        int[] b= new int[10];
        for(int i=0;i<10;i++){
            b[i]=(int)(Math.random()*(max-min+1)+min);
        }
        System.out.println("Generated Array: "+Arrays.toString(b));
        m.heap_sort(b);
        System.out.println("Sorted Generated Array:
"+Arrays.toString(b));
        }

        /*

        Question#4: The average running time of heap sort when n=1000
with 100 iterations is approximately 93341 nanoseconds



        Questions#5: The average running time of heap sort when n=1000
with 100 iterations is approximately 93341 nanoseconds
                     The average running time of selection sort when
n=1000 with 100 iterations is approximately 233287 nanoseconds

                     We can see that the average running time of heap
sort is much smaller than the average running time of selection sort.
                     This idea is also expressed when we take a look
at the implementation of heap sort and selection sort.
                     The time complexity of heap sort is nlog(n) while
the time complexity of selection sort is n^2

        */
```

```
    public void build_MaxHeap(int[] a){
        int n=a.length;

        //using i=n:-1
        for(int i=n-1;i>=0;i--){
            max_heapify(a,i,n);
            //System.out.println(Arrays.toString(a));
        }

        //System.out.println(Arrays.toString(a));

        /*
        We can start at n/2 instead of n because our max_heapify start
with left and right is  2*i and 2*i+1
        which mean if we use the total length (n) of the array, the
max_heapify will not increment left and right
        since they are larger than the total length of the heap.
Hence, to use n instead of n/2 does not change anything
        except the max_heapify will do more unnecessary steps;

        If we use n/2, the left and right will start at an index less
than n and the right will start at n+1. This mean the max_heapify can
now
        start swap the value in the array to make it become a heap
instead of going through a long unnecessary process like with i=n;
        */

        //using i=n/2:-1
        /*for(int i=n/2;i>=0;i--){
            max_heapify(a,i,n);
            //System.out.println(Arrays.toString(a));
        }*/
    }

    public void heap_sort(int[] a){
        build_MaxHeap(a);
        int n=a.length;
        for(int i=n-1;i>0;i--){// remove a[0]
            //System.out.println(a[0]);
            //System.out.println(i);
            arraySwap(a,0,i);
            //System.out.println(Arrays.toString(a));
            max_heapify(a,0,i);
            //System.out.println(Arrays.toString(a));
        }
```

```
    }

    public void max_heapify(int[] a, int i, int n){
        int mx=i;
        int left=2*i;
        int right=2*i+1;

        if( left < n && a[left]>a[mx]){
            mx=left;
        }
        if(right < n && a[right] >a[mx]){
            mx=right;
        }

        if(mx!=i){
            arraySwap(a,mx,i);
            max_heapify(a,mx, n);
        }
    }

    public void arraySwap(int[] a, int b, int c){
        int temp= a[b];
        a[b]=a[c];
        a[c]=temp;
    }

    public void selectionSort(int[] a){
        int n=a.length;
        int currentMinIndex=0;
        //System.out.println(Arrays.toString(a));
        for(int i=0;i<n-1;i++){
            //System.out.println(a[i]);
            currentMinIndex=i;
            for(int j=i+1;j<n;j++){
                if(a[j]<a[currentMinIndex]){
                    //System.out.println( a[j]+" |
"+a[currentMinIndex]);
                    currentMinIndex=j;
                }
            }
            arraySwap(a,i,currentMinIndex);
            //System.out.println(Arrays.toString(a));
        }
    }
}
```