HOMEWORK 1

Problem1:

1-

function	Big oh	order
4n log n + 2n	O (n log n)	6
2^{10}	O (1)	1
$2^{\log n} = n^{\log 2} = n$	O (n)	2
3n+100log n	O (n)	3
4n	O (n)	4
2^n	O (2^n)	9
n^2+10n	O (n^2)	7
n^3	O (n^3(	8
n log n	O (n log n)	5

2-

$\log n^{2n} + n^2 \leq cn^2 \quad \forall n \geq n_0$

$]logn^{2n} \leq c^2 - n^2$

$logn^{2n} \leq (c-1)n^2$

$\frac{logn^{2n}}{(c-1)} \leq n^2$

$which\ is\ true\ for\ c = 2\ when\ n_0 = 4$

3-

$\sum_{i=1}^{5} i^3 \leq c(1)$

$(n(n+1)/2)^2 = 225 \leq c$

$which\ is\ true\ for\ c = 225\ when\ n_0 = 1$

4-

$\sum_{i=1}^{n} \lceil \log i \rceil = \log(1) + \log(2) + \log(3) + \cdots + \log(n-1) + \log(n)$

From Logarithm rules: log (ac) = log (a) + log (c)

$\sum_{i=1}^{n} \lceil \log i \rceil = \log(1 * 2 * 3 \ldots * (n-1) * n) = \log(n!) \leq c(nlogn)$

Which is true for c=1 when n_0 =1

5-

$10n + 5logn \leq cn$

$5logn \leq (c - 10)n$

$\frac{5logn}{c-10} \leq n$

Which is true for c=15 when $n_0 = 1$

Another solution:

$10n + 5logn \leq (10 + 5)n$

$10n + 5logn \leq 15n$

For c=15 when $n_0 = 1$

Problem2:

1-

Since the algorithm chooses log n elements, and each element needs O(n) time calculation, so the worst case is: **O(n log n)**, where **log n** represent the number of elements, and **n** represent the time calculation for each element.

2-

The worst case when all array elements are even, which is **$O(n^2)$,** where **n** represent the number of even elements, and the **other n** represent the time calculation for each even number.

3-

Since algorithm D calls algorithm E for each element in the array, the time computation will be the summation of $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{n^2-n}{2}$ which gave us the worst case with **$O(n^2)$.**

Problem3:

	statements	S/E	freq	total
1	public void func1 (int n) {	0	0	0
2	for (int i = 0; i < n * log(n); i++) {	1	nlogn+1	nlogn+1
3	System . out . println (i);	1	nlogn	nlogn
4	for (int j = 2; j < n; j++) {	1	nlogn(n-2+1)	n^2logn-nlogn
5	System .out . println (j);	1	nlogn(n-2)	n^2logn-2nlogn
6	}	0	0	0
7	}	0	0	0
8	System .out . println (" Goodbye !");	1	1	1
9	}	0	0	0
	Total operation	$2n^2$logn-nlogn+2		
	Big oh	O(n^2logn)		

	statements	S/E	freq	total
1	public void func2 (int n) {	0	0	0
2	for (int i = 0; i < n * n; i++) {	1	n^2-0+1	n^2+1
3	System . out . println (i);	1	n^2	n^2
4	for (int j = 2 * n; j > n; j --) {	1	n^2(2n—n+1)	$2n^3$-n^3+n^2 = n^3+n^2
5	System .out . println (j);	1	n^2(2n-n)	$2n^3$-n^3= n^3
6	}	0	0	0
7	}	0	0	0
8	System . out . println (" Goodbye !");	1	1	1
9	}	0	0	0
	Total operation	$2n^3$+$3n^2$+2		
	Big oh	O(n^3)		

	statements	S/E	freq	total
1	public void func3 (int n) {	0	0	0
2	for (int i = n; i > 0; i --) {	1	n-0+1	n+1
3	System . out . println (i);	1	n	n
4	for (int j = 0; j < i; j++) {	1	$\frac{n(n+1)}{2}+n$	$\frac{n^2+n}{2}+n$
5	System .out . println (j);	1	$\frac{n(n+1)}{2}$	$\frac{n^2+n}{2}$
6	}	0	0	0
7	}	0	0	0
8	System .out . println (" Goodbye !");	1	1	1
9	}	0	0	0
	Total operation	n^2+4n+2		
	Big oh	O(n^2)		

	statements	S/E	freq	total
1	void func4 (int n) {	0	0	0
2	int m = 1;	1	1	1
3	while (m <= n) {	1	n-1+1+1	n+1
4	system . out . println (m);	1	n	n
5	i = n;	1	n	n
6	while (i > 0) {	0	n(logn+2)	nlogn+2n
7	system .out . println (i);	0	n(logn+1)	nlogn+n
8	i = i / 2;	1	n(logn+1)	nlogn+n
9	}	0	0	0
10	m++;		n	n
11	}		0	0
12	}		0	0
	Total operation	3nlogn+8n+2		
	Big oh	O(nlogn)		

Problem3:

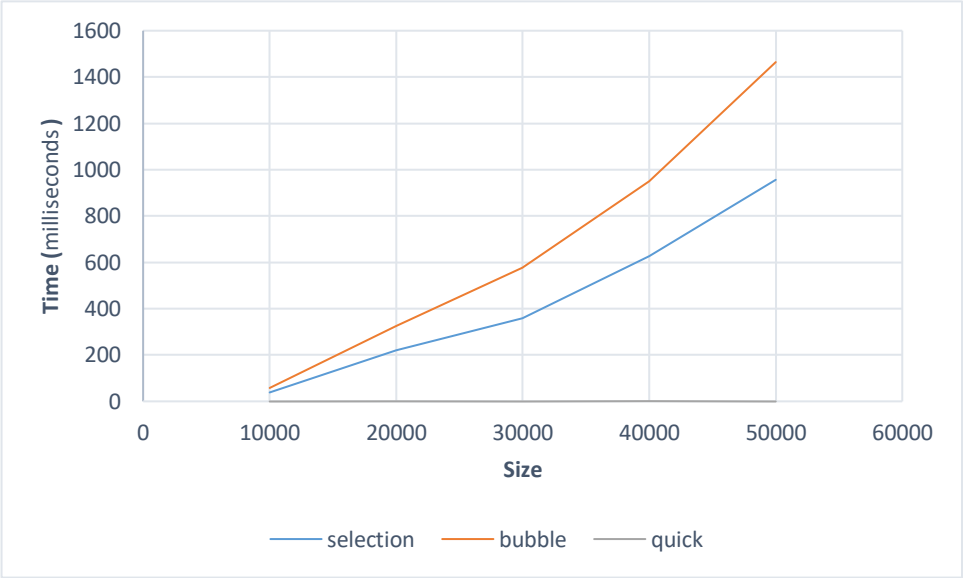
1-

```
public class Test {
   public static void main (String [] args){

      long selection,bubble,quick, time1,time2;


      for (int i=10000; i<= 50000; i=i+10000){

         double[] selectionSort= new double[i];
         double[] bubbleSort = new double [i];
         double[] quickSort = new double [i];
         selection=0;
         bubble=0;
         quick=0;

         for(int j=0; j<i; j++){

            selectionSort[j]=bubbleSort[j]=quickSort[j]=Math.random();;
         }

         for (int x=0 ; x<100 ; x++){
            time1=System.nanoTime();
            Sort.selectionSort(selectionSort,i);
            time2= System.nanoTime();
            selection = selection+(time2-time1);

            time1=System.nanoTime();
            Sort.bubbleSort(bubbleSort,i);
            time2= System.nanoTime();
            bubble = bubble+(time2-time1);

            time1=System.nanoTime();
            Sort.quickSort(quickSort,i);
            time2= System.nanoTime();
            quick = quick +(time2-time1);
         }
```

```
        System.out.println("Averge of selection with size:" + i+ " is: " +
((double)selection/100)/1000000);
        System.out.println("Averge of bubbleSort with size:" + i+ " is: " +
((double)bubble/100)/1000000);
        System.out.println("Averge of quick with size:" + i+ " is: " +
((double)quick/100)/1000000);

      }
   }
}
```

2-

Size\sort type	Selection sort	Bubble sort	Quick sort
10000	38.5	57.79	0.11
20000	220.46	324.78	0.09
30000	359.51	575.7	0.12
40000	625.48	951.5	0.9
50000	956.38	1464.28	0.13



3-

Quicksort is the fastest since it takes the shortest time to sort the array.

4-

Selection sort is faster since it takes shorter time to sort the array with different sizes, bubble sort has larger growth rate than the selection sort because bubble sort take longer time to sort the array, even they have the same time complexity with big oh (n^2) .