Problem 1

1) $5n^2+2n+1 \leqslant 5n^2+2n^2+n^2 \quad n_0=1$
$\leqslant 8n^2 \quad n_0=1$
$O(n^2) \quad c=2 \quad n_0=1$

2) $n^2+nlog(n) \rightarrow O(n^2)$

Prove:
$n^2+nlog(n) \leqslant n^2+n^2 \quad n_0=1$
$\leqslant 2n^2 \quad n_0=1$
$O(n^2) \quad c=2 \quad n_0=1$

3) $2n^3 \notin O(n^2)$
$f(n)=O(n^3)$
$g(n)=O(n^2)$
$f(n) \neq g(n)$

4) a) $0.1n^2$
b) n^2

5) a) True.
b) False, because $100n^3+8n^2+5n$ has $O(n^3)$ which has ranking greater than $O(n^2 \log n)$

6) $\log_a(n) \in O(\log_b(n)) \quad a,b>0$ is false,
for example let a = 2 , b = 10 and n = 1000
$\log_2(1000) \approx 10$
$\log_{10}(1000) \approx 3$

7) $a^n \notin O(b^n) \quad a>b>0$ true,
for example let a = 3, b = 2 and n =10
$3^{10}=1000$
$2^{10}=100$

Problem 2

1)

	Statement	S/E	Freq.	Total
1	int sum = 0	1	1	1
2	for(int i = 1 ; i <= n ; i++)	1	n+1	n+1
3	for int j = 0 ; j < 2 * i ; j++)	1	$\frac{(n+1)^2}{4}-1$	$\frac{(n+1)^2}{4}-1$
4	sum += j	1	$\frac{n(n+2)}{4}-1$	$\frac{n(n+2)}{4}-1$
5	return sum;	1	1	1
			Total	$\frac{3n^2+10n+1}{4}=$ $3/4n^2+10/4n+1/4$ $O(n^2)$

2)

	Statement	S/E	Freq.	Total
1	for(int i = 0; i < n * n * n ; i++)	1	n^3+1	n^3+1
2	System.out.println(i);	1	n^3	n^3
3	for(int j = 2 ; j < n ; j++) {	1	$n^3(n-2+1)$	n^4-n^3
4	System.out.println(j); }	1	$n^4-2n^3+n^3-1$	n^4-n^3-1
5	System.out.println("End");	1	1	1
			Total	$2n^4+1$ $O(n^4)$

3)

	Statement	S/E	Freq.	Total
1	int k = 100, sum = 0;	1	1	1
2	for(int i = 0; i < n ; i++)	1	n+1	n+1
3	for(int j = 1 ; j <= k ; j++) {	1	100n	100n
4	sum = i + j;	1	99n	99n
5	System.out.println(sum);	1	99n	99n
6	}	0	0	-
			Total	299n+2 $O(n)$

Problem3

1. $O(n \log(n))$
2. Best case all number odd $O(n \log(n))$. Worst case all number even $O(n^2)$
3. a- $O(n)$
   b- $O(n)$
   c- $O(n^2)$
   d- $O(n^2)$
   e- $O(n^3)$

Problem 4

1. Best running time is to have the values sorted ascending.
   For example if we have n=8 and array values {1,2,3,4,5,6,7,8} the inner for (k) will not run (0).
   Worst running time is to have the values sorted descending.
   For example if we have n = 8 and array values {8,7,6,5,4,3,2,1}
   the inner for (k) will run 20 times which is $\frac{n(n+2)}{4}$
2. Best case

	Statement	S/E	Freq.	Total
1	public static int func1(int[] A, int n)	0	0	-
2	{	0	0	-
3	int i = 0;	1	1	1
4	int j = n - 1;	1	1	1
5	int sum = 0;	1	1	1
6	int loop = 1;	1	1	1
7	while (i <= j)	1	n+1	n+1
8	{	0	0	-
9	if (A[i] > A[j])	1	n	n
10	{	0	0	-
11	for (int k = i; k <= j; k++)	1	0	0
12	{	0	0	-
13	System.*out*.println(loop++);	1	0	0
14	sum += A[k];	1	0	0
15	}	0	0	-
16	}	0	0	-
17	i++;	1	n	n
18	j--;	1	n	n
19	}	0	0	-
20	return sum;	1	1	1
21	}	0	0	-

Worst case

	Statement	S/E	Freq.	Total
1	public static int func1(int[] A, int n)	0	0	-
2	{	0	0	-
3	int i = 0;	1	1	1
4	int j = n - 1;	1	1	1
5	int sum = 0;	1	1	1
6	int loop = 1;	1	1	1
7	while (i <= j)	1	n+1	n+1
8	{	0	0	-
9	if (A[i] > A[j])	1	n	n
10	{	0	0	-
11	for (int k = i; k <= j; k++)	1	n(n+2)/4	n(n+2)/4
12	{	0	0	-
13	System.*out*.println(loop++);	1	n(n+1)^2/4	n(n+1)^2/4
14	sum += A[k];	1	n(n+1)^2/4	n(n+1)^2/4
15	}	0	0	-
16	}	0	0	-
17	i++;	1	n	n
18	j--;	1	n	n
19	}	0	0	-
20	return sum;	1	1	1
21	}	0	0	-

Problem 5

Space: $O(n^2)$

Problem 6)

1) Code

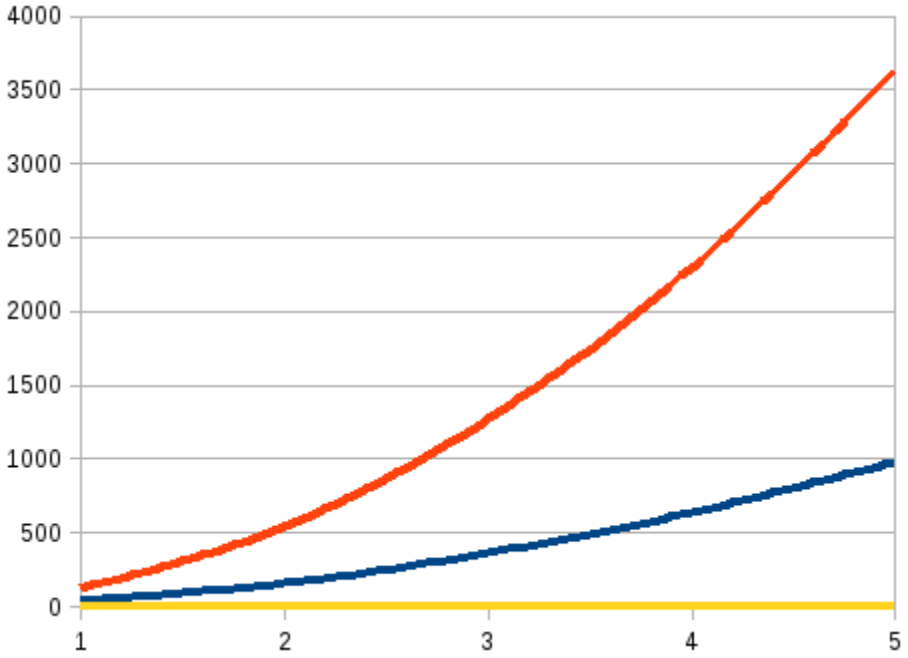
```
import java.util.*;
public class Main
{
	public static void fill(double[] A , int n)
	{
		for (int i = 0; i < n; i++)
		{
			A[i] = Math.random();
		}
}

public static double expSelectionSort (double[] A , int n)
{
	double total = 0;
	for (int i = 0; i < 100; i++)
	{
		double[] b = Arrays.copyOf(A, A.length);
		long befor = System.nanoTime();
		Sort.selectionSort(b, n);
		long after = System.nanoTime();
		total = total + (after - befor);
	}

	total /= 1000000;
	return total / 100;
}

public static double expBubbleSort (double[] A , int n)
{
	double total = 0;

	for (int i = 0; i < 100; i++)
	{
		double[] b = Arrays.copyOf(A,A.length);
		long befor = System.nanoTime();
		Sort.bubbleSort(b, n);;
		long after = System.nanoTime();
		total = total + (after - befor);
	}

	total /= 1000000;
	return total / 100;
}
```

```
public static double expQuickSort (double[] A , int n)
{
	double total = 0;
	for (int i = 0; i < 100; i++)
	{
		double[] b = Arrays.copyOf(A, A.length);
		long befor = System.nanoTime();
		Sort.quickSort(b, n);;
		long after = System.nanoTime();
		total = total + (after - befor);
	}

	total /= 1000000;
	return total / 100;
}

public static void main (String[] args)
{
	int[] data = new int[5];
	double A[] = new double[500000];
	fill(A, 500000);
	System.out.println("Data \t\t SelectionTime\t\t BubbleTime \t\t Quick Time ");
	for (int i = 0 ; i < 5 ; i++)
	{
		data[i] = (i+1)*10000;
		System.out.println(data[i] + " \t\t " + expSelectionSort(A, data[i]) + " \t\t "
		+expBubbleSort(A, data[i]) + " \t\t " + expQuickSort(A,data[i]));
	}
}
}
```

2) The table :

Data	Selection Sort	Bubble Sort	Quick Sort
10000	40.39	126.86	0.88
20000	157.64	543.86	1.38
30000	363.28	1272.34	2.00
40000	636.68	2295.32	2.58
50000	979.74	3630.29	3.34

The graph :



3) The Quick sort is the fastest one.
4) Selection sort is faster than Bubble sort
Bubble sort has a larger growth rate