

Tentamen – extra del: lösning

Uppgifter: lösningar

Uppgift 1 (4 poäng + 3 poäng)

a) (4 poäng)

```
public static void sort (int[] numbers)
{
    int    lastPos = numbers.length - 1;
    int    minPos = 0;
    for (int currentPos = 0; currentPos < lastPos; currentPos++)
    {
        minPos = currentPos;
        for (int p = currentPos + 1; p <= lastPos; p++)
            if (numbers[p] < numbers[minPos])
                minPos = p;

        int    n = numbers[currentPos];
        numbers[currentPos] = numbers[minPos];
        numbers[minPos] = n;
    }
}
```

b) (3 poäng)

I värsta fall utförs precis 1 elementutbyte i varje pass genom huvudloopen. Det finns $n - 1$ pass, och därmed är det totala antalet utbyten:

$$n - 1$$

Det betyder att algoritmens tidskomplexitet i värsta fall, när det gäller antalet elementutbyten, kan ges med följande komplexitetsfunktion:

$$w(n) = n - 1$$

$$w(n) \in \Theta(n)$$

Uppgift 2 (3 poäng)

Definition: mängden $\Theta(n^2)$

En komplexitetsfunktion $f(n)$ tillhör mängden $\Theta(n^2)$ om och bara om det finns två reella, positiva konstanter c_1 och c_2 , och ett icke negativt heltal N , för vilka följande olikheter gäller för alla $n \geq N$:

$$c_1 n^2 \leq f(n) \leq c_2 n^2$$

Uppgift 3 (4 poäng + 3 poäng + 3 poäng)

a) (4 poäng)

```
// add lägger till ett givet element i vektorn
public void add (E e)
{
    if (!(countElements < elements.length))
    {
        Object[]    newElements = new Object[2 * elements.length];
        for (int pos = 0; pos < elements.length; pos++)
            newElements[pos] = elements[pos];
    }
}
```

```
        elements = newElements;
    }

    elements[countElements++] = e;
}
```

b) (3 poäng)

```
// get returnerar det element som finns på en given position
public E get (int index)
{
    return (E) elements[index];
}
```

c) (3 poäng)

