5.31

```
	public static boolean isPrime(int n)
	{
		for (int i = 2; i <= (int) Math.sqrt(n); i++)
		{
			if (n % i == 0)
			{
				return false;
			}
		}
		return true;
	}
```

a) O($\sqrt{N}$)

b) $B \approx lgN$

c) $\sqrt{2^B}$

d) 20 bit: 2^{19}; 40 bit: 2^{39}

5.35

See programs

8.1 a)

```
	8 1 4 1 5 9 2 6 5
	1 8 4 1 5 9 2 6 5
	1 4 8 1 5 9 2 6 5
	1 1 4 8 5 9 2 6 5
	1 1 4 5 8 9 2 6 5
	1 1 4 5 8 9 2 6 5
	1 1 2 4 5 8 9 6 5
	1 1 2 4 5 6 8 9 5
	1 1 2 4 5 5 6 8 9
```

8.4 a) O(N)

8.5 a) O(N)

8.6 a) O(N^2)

8.21

a)

```
public static boolean sumPossible(int[] array, int K)
	{
		int N = array.length;
		for (int i = 0; i < N; i++)
		{
```

```
                        for (int j = 0; j < N; j++)
                        {
                                if (i + j == K)
                                {
                                        return true;
                                }
                        }
                }
                return false;
        }
```

b)

```
public static boolean sumPossible2(int[] array, int K)
        {
                int N = array.length;
                Arrays.sort(array);
                int high = N -1;
                int low = 0;
                int sum;
                while (low <= high)
                {
                        sum = array[low] + array[high];
                        if (sum == K) {return true;}
                        if (sum < K) {low++;}
                        if (sum > K) {high--;}
                }
                return false;
        }
```