Exercise 3.1  Comparison of sorting algorithms.

Let $A[1..n]$ be an array. Consider the following naive implementations of the sorting algorithms bubble sort, insertion sort, selection sort, and quicksort. These algorithms are called with the parameters $l = 1$ and $r = n$ to sort the array $A$ in ascending order.

```java
public void bubbleSort(int[] A, int l, int r) {
    for (int i=r; i>l; i--)
        for (int j=l; j<i; j++)
            if (A[j]>A[j+1])
                swap(A, j, j+1);
}

public void selectionSort(int[] A, int l, int r) {
    for (int i=l; i<r; i++) {
        int minJ = i;
        for (int j=i+1; j<=r; j++)
            if (A[j]<A[minJ])
                minJ = j;
        if (minJ != i)
            swap(A, i, minJ);
    }
}

public void insertionSort(int[] A, int l, int r) {
    for (int i=l; i<=r; i++)
        for (int j=i-1; j>=l && A[j]>A[j+1]; j--)
            swap(A, j, j+1);
}

public void quicksort(int[] A, int l, int r) {
    if (l<r) {
        int i=l+1, j=r;
        do {
            while (i<j && A[i]<=A[l]) i++;
            while (i<=j && A[j]>=A[l]) j--;
            if (i<j) swap(A, i, j);
        } while (i<j);
        swap(A, l, j);
        quicksort(A, l, j-1);
        quicksort(A, j+1, r);
    }
}
```

The function $\text{swap}(A, i, j)$ exchanges (swaps) the elements $A[i]$ and $A[j]$. For each of the above algorithms, estimate asymptotically both the minimum and the maximum number of performed swaps and comparisons of elements of $A$. For each of these cases, give an example sequence of the numbers $1, 2, ..., n$ for which the particular case occurs. The sequence should be described in such a way that any $n$ can be chosen (the descending sorted sequence can be described as $n, n-1, ..., 2, 1$). Enter your results in a table of the following form.

<table>
<thead>
<tr>
<th></th>
<th>bubbleSort</th>
<th>insertionSort</th>
<th>selectionSort</th>
<th>quicksort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparisons</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Permutations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exercise 3.2  Algorithm design: sums of numbers.

Let $A[1..n]$ be an array of natural numbers. For each of the following problems, provide an algorithm that is as efficient as possible, and determine its running time in the worst case.

a) Given a natural number $z$. Does the array $A$ contain any two entries $a$ and $b$ such that $a + b = z$?

b) Suppose that $A$ is sorted in ascending order. How efficiently can the problem from a) be solved now? Hint: In this case it is possible to achieve a better running time than in the previous case.

c) Does the array $A$ contain any three different entries $a$, $b$ and $c$ such that $a + b = c$?

Exercise 3.3  Open addressing.

a) Add the keys 16, 6, 24, 41, 17, 38, and 28 in this order into the hash table below. Use open addressing with the hash function $h(k) = k \mod 11$ and resolve the conflicts using
   (i) linear probing
   (ii) quadratic probing
   (iii) double hashing with $h'(k) = 1 + (k \mod 9)$

   For each of the cases, determine the number of collisions. Which variant is the best for the above situation/input?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

b) Provide a sequence of insert operations such that quadratic probing causes more collisions than linear probing. Specify a rule that prescribes how to form such a sequence of arbitrary length $n$. Use the hash function $h(k) = k \mod m$ for a prime number $m$ with $m \geq n$.