Exercise 2.1  Recurrence Relations.

Find a closed form for recurrence relations of the form

\[ T(n) = \begin{cases} 
  aT\left(\frac{n}{b}\right) + cn + d & \text{falls } n > 1 \\
  e & \text{falls } n = 1 
\end{cases} \]

with \(a, b, c, d, e \in \mathbb{N}, b > 1\). Prove your answer using mathematical induction. You can assume that \(n\) is a power of \(b\).

Note: In your proof, distinguish the cases i) \(a \neq b, a \neq 1\) ii) \(a \neq b, a = 1\) and iii) \(a = b\).

Exercise 2.2  Estimating asymptotic Running Time.

Specify (as concisely as possible) the asymptotic running time of the following code fragments in \(\Theta\) notation depending on \(n \in \mathbb{N}\). You do not need to justify your answer.

```c
1 for(int i = 1; i <= n; i += 10) {
2     for(int j = 1; j <= n/2; j += 4)
3         ;
4 }
```

```c
1 for(int i = 1; i <= n; i++) {
2     for(int j = 1; j*j <= n; j++)
3         ;
4     for(int k = n; k >= 2; k /= 2)
5         ;
6 }
```

```c
1 int f(int n) {
2     if(n <= 1) { return 1; }
3     else { return f(n/2)+f(n/2); }
4 }
```

Please turn over.
Exercise 2.3  Cost Models.

Consider the following code fragment.

```c
1   x=2; k=1;
2   while(k<=n) { x=x*x; k=k+1; }
```

a) Which value does the variable $x$ have (in dependency of $n$) after the above code fragment terminated?

b) Show that the cost of all elementary operations (assignments, arithmetic operations, comparisons) is linear in the uniform cost model.

c) Show that the cost of all elementary operations is at least $2^n$ in the logarithmic cost model.

Exercise 2.4  Algorithm Design: Divide-and-Conquer.

We will say that an array of $n$ elements $A[1..n]$ has a majority element if more than half of his entries are equal. Design an algorithm that decides whether a given array has a majority element, and that also finds the corresponding entry. You cannot assume that the entries come from an ordered universe, so the operator “$<$” cannot be used. You can, however, decide in constant time whether two entries are the same or not.

Show how this problem can be solved in time $O(n \log n)$ using the divide-and-conquer paradigm.

Note: If the array is split in two equal parts, what can you infer from the partial solutions that helps towards the overall solution?