Homework Sheet 8

Author	**Matriculation Number**	**Tutor**
Abdullah Oğuz Topçuoğlu	7063561	Maryna Dernovaia
Ahmed Waleed Ahmed Badawy Shora	7069708	Jan-Hendrik Gindorf
Yousef Mostafa Farouk Farag	7073030	Thorben Johr

Exercise 3

```
void Permute(A[1..n])
   if (n == 1) return
   int randomIndex = rand(n) // rand() function from the lecture
   swap(A[1], A[randomIndex])
   Permute(A[2..n])
```

Correctness Proof:
We will prove by induction that the algorithm produces a uniform random permutation of the array A[1..n].
Base Case: For n = 1, there is only one permutation possible, which is the array itself. The algorithm correctly returns A[1..1].
Inductive Step: Assume that the algorithm produces a uniform random permutation for arrays of size k - 1. We need to show that it also works for an array of size k.
The algorithm selects a random index from 1 to k and swaps the element at that index with the first element. This means that each of the n elements has an equal probability of being placed in the first position, which is 1/k.
After placing one element in the first position, the algorithm recursively permutes the remaining k-1 elements. By the inductive hypothesis, the recursive call produces a uniform random permutation of the remaining elements.
Therefore, by induction, the algorithm produces a uniform random permutation for any array of size n.

Running Time Analysis:
The algorithm makes a single swap and then makes a recursive call on an array of size n-1. The time complexity can be expressed as:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(n-1) + 1 \quad \text{for } n > 1 \\ &\Rightarrow T(n) = O(n) \end{aligned}$$