Solution of Exercisesheet 3

Exercise 3-1

Trying some values:

$k = 2$	$a = 27$	$b = 72$	$c = \|b - a\| = 72 - 27 = 45$	$d = 45$	$e = 54$
$k = 3$	$a = 398$	$b = 983$	$c = \|983 - 398\| = 585$	$d = 585$	$e = 558$
$k = 3$	$a = 398$	$b = 938$	$c = \|983 - 398\| = 441$	$d = 441$	$e = 144$
$k = 3$	$a = 321$	$b = 213$	$c = \|321 - 213\| = 108$	$d = 18$	$e = 81$

In the table above one can see that the sum of the digits of d, respectivly e is always a multiple of 9.
The reason for this lies in the construction of the 'pseudorandom generator'. b has the same digits like a only in an other order. To get c we subtract the greater number of those from the smaller, so c is always positive. Then we remove all 0-digits to get d and scramble the letters again for e. The sum of digits didn't change after the computation of c, so we look at c to argue that this sum is always a multiple of 9.
If $a > b$ we compute for the sum of the digits of $c = a - b$: $(a_1 - b_1) + (a_2 - b_2) + ... + (a_n - b_n) = a_1 + a_2 + ... + a_n - (b_1 + b_2 + ... + b_n)$. But this only hold if $a_n > b_n$.
If $a_i < b_i$ then it is $(a_1 - b_1) + (a_2 - b_2) + ... + (a_{i-1} - b_{i-1} - 1) + (a_i - b_i + 10) + ... + (a_n - b_n) = a_1 + a_2 + ... + a_n - (b_1 + b_2 + ... + b_n) + 9$
The sum of the digit of a has to be the same like the sum of the digits from b, because b has the same digits like a only in an other order, so $a_1 + a_2 + ... + a_n - (b_1 + b_2 + ... + b_n) = 0$.
If $a_i < b_i$ holds for y positions the sum of the digits is $9 \cdot y$. a and b have the same digits, so $a_i < b_i$ holds at least for one position, so $y \geq 1$.
This argumentation is the same for $b > a$, so the sum of the digits from c, respectively d or e, is always a multiple from 9.
If now all but the last digit of e are given one can always determine the last digit, because the sum of all digits has to be a multiple of 9. So the described generator does not pass the next-character test.
If the sum of the digits is already 9, the last bit has to be 9 as well, because all 0-digits had erased. If this is not the case, we could not determine if the last digit has to be 0 or 9.

Exercise 3-2

(a) Is G$_a$(s) = G(s)||0 a secure PRG?
No since the last bit is always 0. This bit is not uniformly at random because the probablitiy of that bit being 0 is 100$\%$ instead of 50$\%$.
$\Rightarrow$ G$_a$ is a not a PRG

(b) Is G$_b$(s||b) = G(s)||b where |b| = 1 a secure PRG?
Yes because adding a single random but fixed bit has the probability 50$\%$ being 1 and 50$\%$ being 0 meaning b is uniformly random. b = $\{0, 1\}$ is a pseudorandom generator and the concatination of two pseudorandom number generators is a pseudorandom number generator itself. Also since b is part of the argument G$_b$ is deterministic.
$\Rightarrow$ G$_b$ is a secure PRG

Solution of Exercisesheet 3

(c) Is $G_c(s) = G(s||0)$ a secure PRG?
Assume $G_c(s)$ is not a secure PRG and an adversary $\mathcal{A}$ that can distinguish $G_c(s)$ from a random generator g(s).
We construct $\mathcal{B}$ that invokes $\mathcal{A}$ and answers any query s by asking $s||0$ to its own oracle, concatenating the answers and returning them to $\mathcal{A}$.
$\mathcal{A}$ has to decide if this is a random generator or not. Since G(s) is a secure PRG and therefore its values uniformly distributed it follows that $\mathcal{B}$ is as efficient as $\mathcal{A}$.
Now there are two cases: If G(s||0) is a PRG, $\mathcal{B}$ perfektly simulates $G_c(s)$. In the case when G(s||0) is a random generator, $\mathcal{B}$ also simulates a random generator. It follows, that $\mathcal{B}$ can distinguish whenever $\mathcal{A}$ can. As this would contradict the security of G(s), such an $\mathcal{A}$ can not exist.
$\Rightarrow$ $G_c(s)$ is a secure PRG.

(d) Is $G_d(s) = G(s||0^{|s|})$ a secure PRG?
Assume $G_d(s)$ is not a secure PRG and an adversary A that can distinguish between $G_d(s)$ and a random generator g(s).
We construct $\mathcal{B}$ that $\mathcal{A}$ and adds l(|s|)-times 0 to each string s.
Since G(s) is a secure PRG and therfore its values uniformly distributed it follows that $\mathcal{B}$ is as efficient as $\mathcal{A}$.
$\Rightarrow$ $G_d(s)$ is a secure PRG.

(e) Is $G_e(s) = G(s) \oplus 1^{l(|s|)}$ a secure PG?
Assume $G_e(s)$ is not a secure PRG and an adversary A that can distinguish between $G_e(s)$ and a random generator g(s).
We construct $\mathcal{B}$ that calls the function $F(s) = s \oplus 1^{l(|s|)} = G(s)$ for $s = G_e(s)$. For $\mathcal{B}$ to distinguish the one case where b is set correctly from g(s) $\mathcal{B}$ has to distinguish (G(*trunc*(s)) from g(s).
However since G(s) is a secure definition by definition. $\Rightarrow$ contradiction
$\Rightarrow$ $G_f(s)$ is a secure PRG.

(f) Is $G_f(s) = trunc(G(trunc(s)))$ a secure PRG?
where *trunc*(x) for a nonempty string x denotes all but the last bit of x.
(For this part, assume that $l(n) > n + 2$, and ignore the fact that G_f is undefined on input strings of length 1.)

Assume $G_f(s)$ is not a secure PRG and an adversary $\mathcal{A}$ can distinguish between $G_f(s)$ and a random generator g(s).
We construct an adversary $\mathcal{B}$ that evokes $\mathcal{A}$ adds one bit b =$\{0,1\}$ to $G_f(s)$.
For $\mathcal{B}$ to distinguish the one case where b is set correctly from g(s) $\mathcal{B}$ would have to distinguish G(*trunc*(s)) from g(s).
However G(s) is a secure definition by definition. Furthermore $\mathcal{B}$ can't know what value *trunc* has deleted from the string since G(s) is uniformly distributed. $\Rightarrow$ contradiction
$\Rightarrow$ $G_f(s)$ is a secure PRG.