Exercise 1

a)

6	4	5	3	1
a	b	c	d	b
b	c	a	b	c
c	a	b	c	a
d	d	d	a	d

We add one vote of $b \succ c \succ a \succ d$.
In the first round, plurality returns scores of $\{a : 6, b : 5, c : 5, d : 3\}$. By the lexicographical tiebreaking mechanism, this leaves us candidates $\{a, b\}$ for the second round - which yields scores $\{a : 11, b : 8\}$ and as such a wins. This manipulation is trivially minimal.

b)
We run the described algorithm to arrive at a 5-manipulated profile

3	2	3	2
g	c	a	a
f	d	b	b
e	e	c	g
d	f	d	f
c	g	e	e
a	a	f	d
b	b	g	c

In short, this manipulation is exactly optimal. Arranging a and b in the first place is trivially optimal, and the other candidates are arranged inversely to the orderings in the original profile, ensuring equal distribution of points and hence the lowest maximum score.

c)
Lemma 1:
Adding a pair of such voters increases the score of candidate d by exactly $2 \cdot |C| - 2$, while the other scores of other candidates are increased by $|C| - 2$. As such, adding such a pair of voters increases the score of d relative to other candidates by exactly $|C|$.

Now, consider any given election $\mathcal{P} = (C, V)$ where some candidate d does not win, alongside some minimally sized manipulation of size k and a profile $\mathcal{P}' = (C, V')$ induced by it.

Given a winning candidate $a \in f(\mathcal{P})$, we know that $\sigma(a, V) \geq \sigma(d, V)$. Furthermore, we know that $\sigma(a, V') \leq \sigma(d, V')$. As adding a single voter may at most increase the difference in scores between two candidates by $|C|$, we know that $|(\sigma(a, V) - \sigma(d, V)) - (\sigma(d, V') - \sigma(a, V'))| \leq k \cdot |C|$.

Now consider a preference profile $\mathcal{P}'' = (C, V'')$ induced by adding k pairs of voters as described in our algorithm. According to Lemma 1, this increases the difference in scores between a and d by exactly $k \cdot |C|$, i.e. $|(\sigma(a, V) - \sigma(d, V)) - (\sigma(d, V'') - \sigma(a, V''))| = k \cdot |C|$, implying $\sigma(d, V'') \geq \sigma(a, V'')$. Furthermore, according to Lemma 1, the scores of candidates $b \in C \backslash \{d\}$ are increased equally, so by Monotonicity of Addition we know $\sigma(d, V'') \geq \sigma(a, V'') \geq \sigma(b, V'')$, and d wins in this constructed profile.

Therefore, the algorithm is a 2-factor approximation for Borda Coalitional Manipulation.

Exercise 2

a)
The idea behind the Schulze method is that to avoid situations where $a \succ b \succ c$ but $a \prec c$ we consider indirect comparisons - which we calculate by considering paths between candidates, i.e. if $a \succ b \succ c$ then $P[a,b] > P[b,a]$ and $P[b,c] > P[c,b]$ and therefore $P[a,c] > P[c,a]$ and $a \succ c$.

To achieve this, the strength of a path is defined as the strength of the weakest comparison within the path - if a is strongly preferred to b, and b strongly preferred to c, a is indirectly strongly preferred to c - but if b is only weakly preferred to c, the connection must of course be weaker, so a is indirectly only weakly preferred to c.

As potentially many paths can be constructed between two candidates, their comparison under the Schulze method depends on the strengths of the strongest paths - if the strongest path from a to b is stronger than the strongest path from b to a, a is considered indirectly preferred to b.

Then, a candidate that no other candidates are indirectly preferred to is the winner.

The Schulze method is computed via dynamic programming - for candidates i, j, k with $i \neq j \neq k$, if a pair of paths from j to i $P_D[j,i]$ and from i to k $P_D[i,k]$ stronger than the known path from j to k $P_D[j,k]$ is found, a new path $P'_D[j,k] > P_D[j,k]$ is constructed from them and saved. Instead of storing paths, predecessor nodes are stored and then recursively retrieved.

With all of the strongest paths being computed as such, we once again iterate over candidates i, j with $i \neq j$ - if the stored path from i to j is stronger than the one from j to i, we know i is preferred to j...

b)
See figures 2b.1 and 2b.2. From the path info in 2b.2 we know that, $\mathcal{O} = \{ab, ca, da, cb, db, cd\}$. As c is the only candidate where $\forall x \in C : xc \notin \mathcal{O}$, we know that $S = \{c\}$.

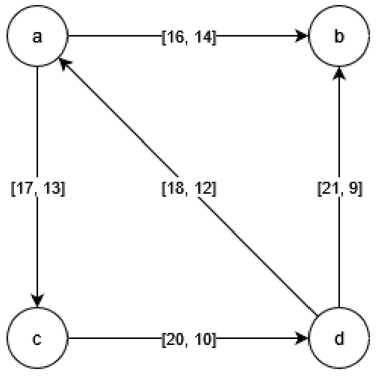

Fig 2b.1: Digraph induced by profile

	a	b	c	d
a	-	16	17	17
b	-12	-	-10	-9
c	18	20	-	20
d	18	21	17	-

Fig 2b.2: Strongest Paths

Exercise 3

a)
In short, independence of clones prevents negative impacts caused by the spreading of votes between highly similar candidates - whether naturally occurring or maliciously induced.
Independence of clones for example prevents two issues commonly observed in Plurality and Borda voting (a) in plurality voting, adding clones of a winning candidate will split the votes between clones - unless the winner has 100% of the votes, there is some number of clones that may be added to have this winner instead lose.
Similarly, (b) in borda voting, adding a somewhat less popular clone b of a candidate a necessarily increases the total score of a by some integer c and the score of the winning candidate by some integer $d \leq c$ - therefore, stacking up such clones can allow a previously unpopular candidate to win.

b)
It was called Maximin voting.

c)
Consider profile 3.1, in which d trivially wins.
Now, consider profile 3.2, which is constructed from profile 3.1 by replacing d with a group of clones a, b, c. In profile 3.2, w loses all pairwise comparisons by -3. Meanwhile, a loses the pairwise comparison with c by -5, b loses to a by -9 and c loses to b by -13, leaving d a winner. As such, these profiles violate 4.6.7.

d)
While Simpson-Kramer necessarily considers *all* comparisons, including matchups between clones (which may be arbitrarily bad), the Schulze method selects a path of comparisons, avoiding inter-clone comparisons.

9	6
d	w
w	d

(a) Profile 3.1

6	5	4	5	4	3
a	b	c	w	w	w
b	c	a	a	b	c
c	a	b	b	c	a
w	w	w	c	a	b

(b) Profile 3.2

Exercise 4

a)
The MinMax set is the set of candidates that perform the *least bad* in comparisons with candidates outside of this set. It relates to the independence of clones axiom in so far as that if a candidate d is replaced by some set of clones K, *iff* d is in the MinMax set, will the clones be in the MinMax set.
The mechanism behind this is that by inclusion of the clones in a subgroup B, comparison between these clones is avoided, i.e. $\Gamma_D(B) = \Gamma_D(K \cup B \backslash \{d\}$

b)
The intuition is as such: since a is in the MinMax set and b is not, the best performing subset including a (A) outperforms the best performing subset that includes b and therefore does not include b.

Any path from b to a must include a comparison between a member of A and a non-member - according to the definition of the MinMax set, this is then a terribly weak link, and any such path therefore performs very poorly.

Similarly, a path from a to b must include a link between a member and a non-member of the best performing set including b (B) - and in doing so, path through poorly performing subsets including b. However, the poor performance of these subsets implies at least one member with a strong link from the outside in, so a fairly strong path from a to b can be constructed.

Then, this strong path from a to b ranks above the poor path from b to a.

c)
For $m \in \{0, 1\}$, any profile corresponds to a MinMax set of size m.
For any $m > 1$ we arrange candidates C in a Condorcet Cycle (see Profile 4.1). Then, for any candidates $c_i, c_j, c_k, c_i \neq c_j \neq c_k$ we know $(N[c_i, c_j], N[c_j, c_i]) = (N[c_i, c_k], N[c_k, c_i])$, i.e. the candidates perform the exact same.

Therefore, $\Gamma_D(B)$ is constant over all real subsets B of C. As such, $\Gamma_D(B)$ is minimal for all such subsets, and therefore all such subsets are included in the MiniMax set - which then includes all candidates.

1	1	...	1
c_1	c_n	...	c_2
c_2	c_1	...	c_3
$\vdots$	$\vdots$	...	$\vdots$
c_n	c_{n-1}	...	c_1

(a) Profile 4.1