In the “warmup” algorithm below, $S$ is the set of variables that may or may not be relevant; variables not in $S$ are known to be relevant. $\ell$ is the # of variables that are known to be relevant.

Preliminary Algorithm:

**Naive-Junta-Test**$(f, k, \varepsilon)$

1. Initialize $S = [n]$, $\ell = 0$.
2. For $r = 6(k + 1)/\varepsilon$ rounds, repeat the following:
   - Draw random $x, y \in \{-1, 1\}^n$. If $f(x) \neq f(y_{\overline{S}}x_S)$, then:
     - (a) Use binary search to find a relevant coordinate $j$.
     - (b) Update $S \leftarrow S \setminus \{j\}$, update $\ell \leftarrow \ell + 1$.
     - (c) If $\ell > k$ then halt and output “reject.”
3. If didn’t reject in the $r$ rounds, halt and output “accept.”

**Theorem 1.** Algorithm **Naive-Junta-Test** uses $O(k/\varepsilon + k \log n)$ queries and is a 1-sided $\varepsilon$-tester for whether $f \in J_k$.

The real algorithm:

**Junta-Test**$(f, k, \varepsilon)$

1. Initialize $S = [n]$, $\ell = 0$, $s = \text{poly}(k/\varepsilon)$ (more precisely, $s = 10^{20k^9/\varepsilon^5}$)
2. Randomly partition $[n]$ into $\mathcal{I} = \{I_1, \ldots, I_s\}$ (assign each $i \in [n]$ to uniformly chosen element of $\mathcal{I}$)
3. For $r = 12(k + 1)/\varepsilon$ rounds, repeat the following:
   - Draw random $x, y \in \{-1, 1\}^n$. If $f(x) \neq f(y_{\overline{S}}x_S)$, then:
     - (a) Use binary search to find a set $I_j$ that contains a relevant variable (flip whole block at a time)
     - (b) Update $S \leftarrow S \setminus I_j$, update $\ell \leftarrow \ell + 1$.
     - (c) If $\ell > k$ then halt and output “reject.”
4. If didn’t reject in the $r$ rounds, halt and output “accept.”

**Theorem 2.** Algorithm **Junta-Test** uses $O(k/\varepsilon + k \log k)$ queries and is a 1-sided $\varepsilon$-tester for whether $f \in J_k$. 