Final exam in complexity course.

Please hand in your solutions by February 27, even if you do not solve all questions. You may answer either in Hebrew or in English, either printed or in *clear* handwriting. You may use without proof anything explained in class (including theorems that were stated in class without proof), provided that you state exactly what you use and how you use it. (To avoid misunderstanding let me give an example. Suppose the question is “prove that $3 > 1$”. An acceptable answer is “In class we proved that $3 > 2$, that $2 > 1$, and that the $>$ relation is transitive. It follows that $3 > 1$.” An answer such as “In class we proved that $3 > 2$. The proof that $3 > 1$ is similar.” will not give full credit, unless you also provide the relevant details of the proof.) You may consult the lecture notes, textbooks, your own notes etc., but please do the exam on your own. Note that typically an answer to a question combines more than one topic taught in class. If any question is unclear or appears too difficult, please let me know (the sooner the better).

1. **Randomized space bounded algorithms that surely output the correct answer.** We have seen in class that there is a randomized logspace algorithm that given an undirected graph and two vertices s and t, checks whether s and t are connected (namely, whether there is a path between s and t). If the algorithm answers *yes* then s and t are surely connected. But if the algorithm fails to answer yes, one is not really sure whether s and t are not connected, or whether the algorithm was unlucky in its sequence of coin tosses. Design a randomized logspace algorithm that stops with probability 1 and gives the correct answer regarding whether s and t are connected (both in the case that they are connected and in the case that they are not). On every input the expected running time of the algorithm must be polynomial. Hint: adapt the technique of inductive counting to this setting, e.g., by giving a randomized logspace algorithm that inductively counts the number of connected pairs of vertices in the graph, where the kth step of the induction considers the subgraph induced on the first k vertices of the input graph.

2. **Construct a pseudorandom generator that fools constant depth circuits** (depth 7, for concreteness). More specifically, one needs to construct a family of pseudorandom generators, one for each value of

n. Each pseudorandom generator should be a polynomial size circuit (of size at most n^a, where a is a constant independent of n) that computes some function $f : \{0,1\}^k \longrightarrow \{0,1\}^n$, where $k < n$ is some slowly growing function of n (for example, $k = O((\log n)^2)$). For every constant $b > 0$, and for large enough n (that may depend on b), the following must hold. For every circuit C with n inputs and one output, where C is of size at most n^b and of depth at most 7, the probability that C outputs 1 on an n-bit string that is generated by picking a string in $\{0,1\}^k$ and applying f to it, is the same (up to an additive difference of at most $1/n^b$) as the probability that C outputs 1 on a random input string from $\{0,1\}^n$.

3. **Time space tradeoff for the language of palindromes.** Let P be the language of palindromes. That is, $x = (x_1, ..., x_n) \in P$ iff $(x_1, ..., x_n) = (x_n, ..., x_1)$. Let T be a Turing machine with one read-only input tape and one read/write memory tape of size $s = s(n)$. Let $t = t(n)$ be the running time of T. Show that if T accepts P then $t = \Omega(n^2/s)$. Hint: use communication complexity.

4. **Nondeterministic circuits for graph nonisomorphism.** A nondeterministic circuit $C : \{0,1\}^{n+m} \longrightarrow \{0,1\}$ has two types of inputs. One is the usual input string $x = x_1, \ldots, x_n$, and the other is a *witness* string $w = w_1, \ldots, w_m$. The circuit is said to *accept* an input string x if there is some setting to the witness string w such that $C(x, w) = 1$. Show that the language GNI (graph nonisomorphism, that was discussed in class in the context of interactive proofs) has nondeterministic circuits of size polynomial in n. (The input x encodes a pair of graphs under some standard encoding, such as by giving their adjacency matrices.)