21.- Alice and Bob play scrable often. Alice is a better player, so the probability that she wins any given game is 0.6, independent of all other games. They decide to play a tournament of n games. Bound the probability that Alice loses the tournament, using a Chernoff bound.

22.- (MU 4.6) (*Part (b) is cumbersome, but it is a real application*)

(a) In an election with two candidates using paper ballots, each vote is independently misrecorded with probability $p = 0.02$. Use a Chernoff bound to bound the probability that more than 4% of the votes are misrecorded in an election of 1000000 ballots.

(b) (*Optional*) Assume that a misrecorded ballot always counts as a vote for the other candidate. Suppose that candidate A received 510000 votes and that candidate B received 490000 votes. Use Chernoff bounds to bound the probability that candidate B wins the election owing to misrecorded ballots. Specifically, let X be the number of votes for candidate A that are misrecorded and let Y be the number of votes for candidate B that are misrecorded. Bound $((X > k) \cap (Y > \ell))$ for suitable choices of k and ℓ.

23.- (*) A fundamental problem that arises in many applications is to compute the size of the union of a collection of sets. The setting is the following. We are given m sets $\{S_1, \ldots, S_n\}$ over a very large universe U. The operations we can perform on the sets are the following

(a) size(S_i): returns the number of elements in S_i,

(b) selection(S_i): returns an element of S_i chosen u.a.r.

(c) lowest(x): for some $x \in U$, returns the smallest index i for which $x \in S_i$.

Let $= \cup_{i=1}^{m} S_i$ be the union of the sets S_i. In this problem we will develop a very efficient (polynomial in m) algorithm for estimating the size $|S|$. (We output a number in the range $[(1-\epsilon)|S|, (1-\epsilon)|S|)]$.

(a) A natural example where such a set system arises is the following. Suppose ϕ is a Boolean formula over X variables ($|X| = n$) in *disjunctive normal form* (DNF), i.e. a Boolean formulae $\phi = (C_1 \vee C_2 \vee \cdots \vee C_m)$ where each C_i is a conjunction ($\wedge$) of possibly negated variables. For ex. $(\bar{x}_1 \wedge x_3 \wedge \bar{x}_4) \vee (\bar{x}_1 \wedge x_2 \wedge \bar{x}_3) \vee (\bar{x}_2 \wedge x_4)$. The problem consists in finding an assignment $A : X \to \{0, 1\}$ such that $A(\phi) = 1$. Let U be the set of all possible assignments to the variables of ϕ (i.e. $|U| = 2^n$), and for each clause C_i, $1 \le i \le m$, let S_i denote the set of assignments that satisfy C_i (evaluate C_i to 1). Then the union $S = \cup_{i=1}^m S_i$ is exactly the set of satisfying assignments of ϕ, and our problem is to count them. Argue that all of the above operations can be efficiently implemented for this set system.

(b) Consider a naive random sampling algorithm. Assume that we are able to pick an element of the universe S uniformly at random, and that we know the size $|U|$. Consider an algorithm that picks t elements of U independently and u.a.r. (with replacement), and outputs the value $q|U|$ where t is the proportion of the t sampled elements that belong to S. For the DNF example above, explain as precisely as you can why this is not a good algorithm.

(c) Consider now the following algorithm, which is again based on random sampling but in a more sophisticated way:

- choose a random set S_i with probability $\frac{|S_i|}{\sum_{j=1}^m |S_j|}$
- $x = \text{select}(S_i)$
- if lowest$(x) = i$, then output 1, else output 0

Show that this algorithm outputs 1 with probability exactly $\frac{|S|}{\sum_{j=1}^m |S_j|}$. (Hint: Show that the effect of the first two lines of the algorithm is to select a random element from the set of pairs $\{(x, S_i) | x \in S_i\}$.)

(d) Show that $p \geq 1/m$

(e) Now suppose that we run the above algorithm t times and obtain the sequence of outputs $(X_1, X_2, \ldots, X_t)$. Define $X = \sum_{i=t}^t X_i$. Use Chernoff to obtain a value t (as a function of m, δ, ϵ) that ensures that $\mathbf{Pr}\left[|X - tp| \geq tp\epsilon\right] \leq \delta$. (Hint: need to use the fact $p \geq 1/m$.)

(f) The final output of your algorithm will be $Y = (\sum_{i=1}^m |S_i|) \cdot (X/t)$, where X is defined previously.

Show that this algorithm has the following properties: it runs in time $O(m\epsilon^{-2} \lg(1/\delta))$ (assuming that each of the set operations can be performed in constant time), and outputs a value that is in the range $[(1 - \epsilon)|S|, (1 - \epsilon)|S|)]$, with probability at least $1 - \delta$.