Problem 1

(a) For a formula $\varphi(x_1, \ldots, x_n)$ on n variables $x_1, \ldots, x_n$ and $b \in \{0,1\}^k$ (for $k \leq n$), let φ_b be the formula $\varphi(b_1, \ldots, b_k, x_{k+1}, \ldots, x_n)$. Since C is given as input, we may simulate in polynomial time C on input φ. In particular, in time polynomial in $|C|$ and n we can check that

$$C(\varphi_b) = C(\varphi_{b0}) + C(\varphi_{b1}).$$

Observe that if C satisfies this condition for all partial assignments b (and also, when $k = n$, $C(\text{true}) = 1$, $C(\text{false}) = 0$), then it must be the case that $C(\varphi) = \#\text{SAT}(\varphi)$. This is exactly how we count satisfying assignments recursively. So we have that

$$(C, 1^n) \in L \iff \forall\varphi\forall b : C(\varphi_b) = C(\varphi_{b0}) + C(\varphi_{b1}) \text{ and } C(\text{true}) = 1 \text{ and } C(\text{false}) = 0.$$

and $L \in \text{coNP}$.

(b) By Toda's theorem we know that $\Sigma_2 \subseteq \text{P}^{\#\text{P}}$, so we need only prove that $\text{P}^{\#\text{P}} \subseteq \Sigma_2$. Clearly, it suffices to show that $P^{\#3\text{SAT}} \subseteq \Sigma_2$. Let $L' \in P^{\#3\text{SAT}}$ and a PTM M such that

$$x \in L' \iff M^{\#3\text{SAT}}(x) \text{ accepts.}$$

Now, since $M^{\#3\text{SAT}}(x)$ runs in polynomial time for all x, it can only make polynomially many queries of polynomial length to its oracle. Let C be a circuit of polynomial size that can solve every instance of #3SAT that M may ask on input x (say M makes queries of length up to $p(|x|)$). Then there is a PTM M' such that $M'(C, x) = M^{\#3\text{SAT}}(x)$, and

$$x \in L' \iff \exists C : \ (C, 1^{p(|x|)}) \in L \text{ and } M'(C, x) \text{ accepts.}$$

Since the statement after the first existential quantifier is in coNP—by part (a)—then $L' \in \Sigma_2$.

Problem 2

Suppose $\text{P}^{\#\text{SAT}} \subseteq \text{BPP}^{\text{SAT}}$. By the Gacs-Sipser theorem $\text{BPP} \subseteq \text{NP}^{\text{SAT}}$. This does not directly imply that $\text{BPP}^{\text{SAT}} \subseteq \text{NP}^{\Sigma_2\text{SAT}} = \Sigma_3$. However, if we take a closer look at the proof of the Gacs-Sipser theorem, we will see that the the proof still holds when the BPP machine is augmented with a SAT oracle. (The technical term is that "the proof relativizes", and not all known proofs do.) Therefore, since $\#SAT$ is $\#P$-complete, $P^{\#P} \subseteq \Sigma_3$. By Toda's theorem, for all positive integers k $\Sigma_k \subseteq P^{\#P}$, and the polynomial hierarchy collapses at the third level.

Problem 3

This question turned out fairly difficult so we will discount it in the grading, but let's sketch the solution. Let φ be the formula on n variables whose solutions we want to sample and $S \subseteq \{0,1\}^n$ the set of its satisfying assignments.

One idea is to choose a hash function $h : \{0,1\}^n \to \{0,1\}^k$, hope that there is a unique assignment $x \in S$ such that $h(x) = 0$ and output this assignment. However if h is pairwise independent this does not guarantee that x is uniformly distributed in S.

Instead, we use a hash function with fewer bins: Suppose we chose k such the the expected number of satisfying assignments in every bin is between n^2 and $2n^2$, that is $n^2 \leq |S|/2^k < 2n^2$. (If $|S| < n^2$ we can enumerate all the assignments using the NP oracle and sample from there.) We can now hope that with good probability, there is no bin in which the number of satisfying assignments exceeds this expectation by more than a factor of two, namely

$$\text{For all } b \in \{0,1\}^k,\ |\{x \in S : h(x) = b\}| \leq 4n^2. \tag{1}$$

If this is indeed the case, then we can sample a satisfying assignment as follows: Choose a random pair (i, b), where $1 \leq i \leq 4n^2$ and $b \in \{0,1\}^k$. Using the NP oracle, find a list of all the satisfying assignments x such that $h(x) = b$. If the list has more than i assignments, output the ith assignment in the list, otherwise fail. Conditioned on not failing, the assignment we output is uniformly distributed in S, and since $|S| \geq n^2 \cdot 2^k$, we fail with probability at most $3/4$.

How do we know that we have chosen a correct value for k? We don't, but we can guess one at random between 0 and $n-1$. For a particular value of k, using the NP oracle, we can test whether condition (1) is violated. If it is, we try a different k. If it isn't, we sample as above. If the k is not right, then chances are that either the sampling will fail (if k is too large) or condition (1) will be violated (if k is too small). But even if by some miracle we manage to output a sample, as long as condition (1) holds, the sample will be uniformly distributed in S. We keep trying until we choose the correct k. Each experiment has success probability of at least $1/4n$, so the expected running time of the algorithm is $4n$ trials.

It remains to show that when $n^2 \leq |S|/2^k < 2n^2$, condition (1) holds with probability at least say $1/2$. (This adds a factor of two to the running time of the algorithm.) Unfortunately to show (1) we need more than pairwise independence; we need the hash function family to be n-wise independent, that is

$$\Pr[h(x_1) = y_1 \text{ and } \ldots \text{ and } h(x_n) = y_n] = \Pr[h(x_1) = y_1] \ldots \Pr[h(x_n) = y_n]$$

for all distinct $x_1, \ldots, x_n \in \{0,1\}^n$ and $y_1, \ldots, y_n \in \{0,1\}^k$. Just like pairwise independent random variables satisfy the Chebyshev inequality, n-wise independent variables satisfy the tighter nth moment inequality (when n is even):

$$\Pr[X_1 + \cdots + X_s > \mu + t] \leq 8 \cdot \frac{(\mu n)^{n/2} + n^n}{t^n}. \tag{2}$$

where $\mu = \mathrm{E}[X_1 + \cdots + X_s]$. In our scenario, we fix $b \in \{0,1\}^k$, set $s = |S|$, and let X_i be the indicator of the event that h maps the ith element of S to b. Then $\mu = s/2^k < 2n^2$, so for $t = 2n^2$

we have

$$\Pr[X_1 + \cdots + X_s > 4n^2] \leq 8 \cdot \frac{(2n^3)^{n/2} + n^n}{(2n^2)^n} \leq 2^{-n}.$$

for sufficiently large n. Taking a union bound over k it follows that with probability at least $1/2$, the condition holds simultaneously for all $b \in \{0,1\}^k$.

We sketch a proof of (2). Let $X = X_1 + \cdots + X_s$. By Markov's inequality, when n is even,

$$\Pr[X > \mu + t] = \Pr[(X-\mu)^n > t^n] \leq \mathrm{E}[(X-\mu)^n]/t^n.$$

The value of the expression $\mathrm{E}[(X-\mu)^n]$ is the same whether the X_i are n-wise independent or fully independent; so we will assume they are fully independent. Then we have the Chernoff bound

$$\Pr[X - \mu \geq \lambda] < e^{-\lambda^2/2\mu} + e^{-\lambda/2}$$

and

$$\begin{aligned}
\mathrm{E}[(X-\mu)^n] &= \sum_{\lambda=0}^{\infty} \Pr[(X-\mu)^n \geq \lambda] \\
&= \sum_{\lambda=0}^{\infty} \Pr[X - \mu \geq \lambda^{1/n}] \\
&< \sum_{\lambda=0}^{\infty} e^{-\lambda^{2/n}/2\mu} + e^{-\lambda^{1/n}/2} \\
&\leq \int_0^{\infty} e^{-\lambda^{2/n}/2\mu} d\lambda + \int_0^{\infty} e^{-\lambda^{1/n}/2} d\lambda \\
&= n \cdot (2\mu)^{n/2} \cdot (n/2-1)! + n \cdot 2^n \cdot n!
\end{aligned}$$

The last step uses a change of variables and the convenient identity $\int_0^{\infty} e^{-u} u^{k-1} du = k!$. Using the Stirling bound for the factorial we derive (2).

Problem 4

(a) We can reduce $\#CNF$—which is known to be #P-complete—to $\#DNF$. Given a formula φ in CNF, we first produce $\neg\varphi$. Since $\neg\varphi$ is in DNF and has the same number of clauses as φ, this step of the reduction can be done in polynomial time. Then we count the number s of satisfying assignments for $\neg\varphi$ (using our $\#DNF$ oracle) and we return $2^n - s$. Correctness follows from the fact that an assignment satisfies φ if and only if it doesn't satisfy $\neg\varphi$. (We have shown that $\#CNF \in \mathrm{P}^{\#DNF}$; this reduction is not parsimonious.)

(b) In this problem we refer to the notation used in the hint. We show first how we can sample uniformly from A. To do this, it suffices to know the size of A. We proceed by counting how many assignments satisfy c_i, for each $i = 1, \ldots, m$. This is easy, since we only need to set the literals in c_i to true, and then the rest can take all possible values. Let s_i denote the number

of assignments that satisfy c_i; then $|A| = \sum_{i=1}^{m} s_i$. To sample uniformly from A pick each pair (a, i) with probability $1/|A|$.

Now, those assignments that satisfy a lot of clauses, will be sampled more often than others. What we really want to do, is sample uniformly from B. Of course, we can't calculate the size of B. The algorithm below circumvents this problem.

UNIFORM-DNF(φ)
1 $(a, i) \leftarrow$ uniform pair from A
2 **if** $(a, i) \in B$
3 **then** return a
4 **else goto** 1

We have already discussed step 1 of the algorithm. Step 2 can also be done in polynomial time, since we only have to check that a does not satisfy a clause of $c_1, \ldots, c_{i-1}$. Next we need to argue that every assignment is picked with the same probability. To this end, let t_a be the number of clauses satisfied by a. The probability that an assignment a is picked is as follows.

$$\begin{aligned}
\Pr[a \text{ is picked}] &= \Pr[\text{a pair containing } a \text{ is picked at step 1 and } (a,i) \in B] \\
&= \Pr[\text{a pair containing } a \text{ is picked at step 1}] \cdot \Pr[(a,i) \in B] \\
&= \frac{t_a}{|A|} \cdot \frac{1}{t_a} \\
&= \frac{1}{|A|} \quad (\text{This is independent of } t_a).
\end{aligned}$$

Thus, each assignment is picked with the same probability. Lastly, we need to show that it won't take the algorithm to long before it returns something. This follows from the fact that each loop of the algorithm terminates with probability at least $1/m$ (since for any a, $t_a \leq m$); thus, the expected running time of the algorithm is at most m times the polynomial time needed for steps 1 and 2.