- Start as early as possible, and contact the instructor if you get stuck.
- See the course outline for details about the course's grading policy and rules on collaboration.
- Submit your completed solutions to **Crowdmark**.

1. **Three-sided coin** [**5 marks**]
   Suppose we want to sample uniformly from three possibilities, $\{a, b, c\}$, using a stream of uniformly random bits. For all $B \in \mathbb{N}$, show that if a probabilistic Turing machine always uses fewer than B bits then it cannot sample from the *exact* three outcome distribution.
   (a) Show that if a probabilistic Turing machine *always* uses a fixed number of bits to output a, b, or c, then it cannot sample them uniformly.
   (b) Give an algorithm that samples exactly from the uniform distribution on $\{a, b, c\}$. Analyze the expected runtime of your algorithm.

2. BPPSPACE [**5 marks**]
   Recall from class bounded-error polynomial space.

   $$\mathsf{BPPSPACE} = \{\text{languages } \tfrac{1}{3}\text{-decided by a probabilistic TM in polynomial space}\}$$

   In the time-space hierarchy theorem, we saw that $\mathsf{SPACE}(s) \subseteq \mathsf{TIME}(2^{O(s)})$ because if a machine falls into the same configuration twice, it will loop forever. For a probabilistic Turing machine, it is not so clear, since a random choice may break the machine out of the loop.
   Design a probabilistic Turing machine (high-level steps, not state diagram) that runs for expected *doubly*-exponential time in the length of the input. Justify the correctness and runtime of your TM.[1]

3. **Space constructible and space hierarchy** [**12 marks**]
   We saw the time-hierarchy theorem in class, and there is an analogous space hierarchy theorem:[2]

   Theorem 1. *For all functions $f, g\colon \mathbb{N} \to \mathbb{N}$ such that*
   (a) $\log n \leq f = o(g)$, and
   (b) g is space constructible,
   we have $\mathsf{SPACE}(f) \subsetneq \mathsf{SPACE}(g)$.

   We say a function $h\colon \mathbb{N} \to \mathbb{N}$ is *space constructible* if given $\mathtt{1}^n$ as input, we can construct $\mathtt{1}^{h(n)}$ in space $O(h(n))$, for all $n \in \mathbb{N}$.

   (a) First, argue that for any Turing machine M deciding a language $\mathcal{L}(M)$ in $s(n)$ space, then there exists a Turing machine M' deciding the same language, but using $\Gamma = \{\square, \mathtt{0}, \mathtt{1}\}$ and $\mathcal{O}(s(n))$ space.
   (b) What is the space overhead of simulation? Assume the simulator and the machine being simulated use the same tape alphabet, Γ. Show that there exists an

[1] Nonetheless, it turns out that $\mathsf{PSPACE} = \mathsf{BPPSPACE}$.
[2] Not to be confused with the Time-Space hierarchy theorem we saw in class.

input/working tape Turing machine U which, given $\langle M \rangle x$ as input, simulates M on x in the same space plus $\mathcal{O}(\log n)$, where n is the length of x.

(c) Prove the space hierarchy theorem *carefully*. Point out where you use each hypothesis. We encourage you to use the language

$$A = \{\langle M \rangle x : M \text{ rejects } \langle M \rangle x \text{ in } g(n) \text{ space}\}.$$

(d) Construct a function f such that $f(n) \geq \log n$ and f is not space constructible. Hint: We don't have many tools to show that you *can't* compute a function.

4. **Solar Powered PC [10 marks]**

Suppose L is a language decidable in polynomial space. Let's say we have enough time and space to decide it, but our computer has one major defect: it is solar powered, without a battery or permanent storage. Every night it forgets almost everything; *all* data from memory is lost except for

- the problem input, $x \in \{0,1\}^n$,
- the current day (as a number $d \in \mathbb{N}$),
- 3 bits of output $b \in \{0,1\}^3$, from the previous day.

Surprisingly, we can still decide L with this device. In particular, we will use the three-bit programs from A4Q3, and assume the following theorem.[3]

Theorem 2. *For any polynomial-size family of formulas, we can construct an equivalent polynomial-length family of three-bit programs.*

(a) Recall the TQBF problem, and use it to show that any problem in PSPACE is solved by an exponential-size family of Boolean formulas.

(b) Treat Theorem 2 as a black box (ignore how it was proved), and argue that we can scale it up to exponential-size formulas and exponentially long three-bit programs.

(c) Omitting some details[4], this finishes the proof.
Why doesn't the result contradict our normal intuition (not to mention the space hierarchy theorem) that some problems in PSPACE require polynomial space? Identify the loophole in a couple of sentences.

[3] You do *not* need to have solved A4Q3 for this problem.

[4] We omit the details of (i) computing the length of the program so we know when to stop, and (ii) computing the m'th program instruction on the m'th day.