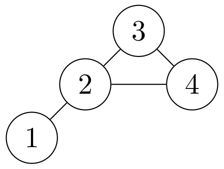
Maximum score: 145 points.
Instructions: For this test, you work in teams to solve multi-part, proof-oriented questions. Problems that use the words "compute," "find," "draw," or "write" require only an answer; no explanation or proof is needed. Unless otherwise stated, all other questions require explanation or proof. The problems are ordered by content, *not difficulty*. The difficulties of the problems are generally indicated by the point values assigned to them; it is to your advantage to attempt problems throughout the test. In your solution for a given problem, you may cite the statements of earlier problems (but not later ones) without additional justification, even if you haven't solved them. Footnotes are not necessary to understand or solve the contents of the round.

1 Graphs and Proofs (45 pts)

Welcome to the power round! We will learn a new form of proof, which you probably haven't heard of. It is a type of proof that lies at the center of modern cryptography and security–the **zero-knowledge proof** (or ZKP).
To begin, let's give some background on problems that ZKPs solve. Consider a group of people at an event, like a math contest, who are friends with some of the other people at the event. If we wanted to represent the relations between these people, we could use a point to represent each person and lines between points to denote that the people represented by the points in question are friends. This makes sense because friendships are symmetric relationships (if A is friends with B, then B is friends with A), and no one is friends with themselves. Such a representation has a name.

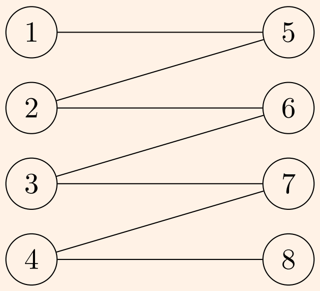
Definition 1.1. A **graph** $G = (V, E)$ is a set of vertices, V, and a set of edges, E. Each edge is itself an (unordered) set of two distinct vertices. Vertices that share an edge between them are called **adjacent**. The number of vertices in a graph is denoted $|V|$ and the number of edges is denoted $|E|$.



In the example above, vertices could represent people at our event, while an edge could indicate that the two people sharing the edge are friends. Formally, one can record this graph with vertex set $V = \{1, 2, 3, 4\}$ and edge set $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{2, 4\}\}$.

Question 1.1 (2 pts)**.** Draw a graph with $|V| = 5$ and $|E| = 7$ and give its vertex and edge sets.

Question 1.2 (2 pts)**.** Find the vertex and edge sets of the following graph.



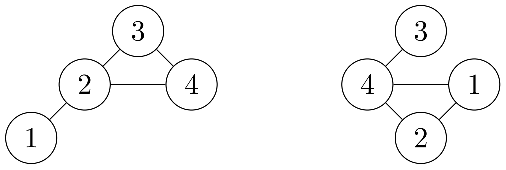
Question 1.3 (2 pts)**.** Draw the graph with the vertex set $V = \{1, 2, 3, 4, 5, 6, 7\}$ and edge set $E = \{\{2, 3\}, \{4, 5\}, \{1, 3\}, \{3, 6\}, \{2, 6\}\}$.

Question 1.4 (3 pts)**.** Suppose a graph has n vertices. Compute the lowest number and highest number of edges it could have.

Graphs can be used for a variety of situations, meaning that different situations could result in similar graphs. For example, a graph to model three people who know each other could be very similar to a graph modeling three cities that all have direct routes to each other. Functionally, these graphs are the same, so we have an important distinction for them.

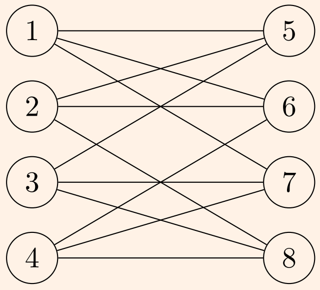
Definition 1.2. An **isomorphism** between graphs G_1 and G_2 is a function f that maps vertices from one graph to another so that edges in G_1 are also represented in G_2, and edges in G_2 all correspond to some edge of G_1. More precisely, f is an invertible function (also called a **bijection**) such that $\{u, v\}$ is an edge of G_1 if and only if $\{f(u), f(v)\}$ is an edge of G_2.
We say graphs G_1, G_2 are **isomorphic** and write $G_1 \cong G_2$ if there's an isomorphism between them.

Intuitively, an isomorphism between two graphs G_1, G_2 just means that one can re-label the vertices of G_1 such that the resulting graph is exactly the graph G_2. Here are two isomorphic graphs.

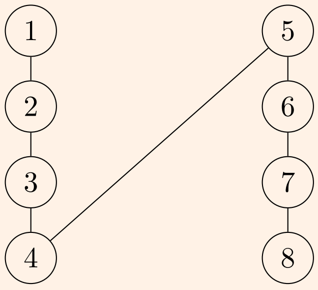


One isomorphism f to get from the left graph to the right graph is given by $f(1) = 3, f(2) = 4, f(3) = 1, f(4) = 2$.

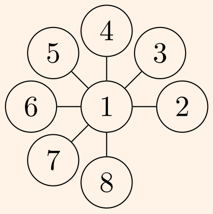
Question 1.5 (4 pts)**.** For the following eight graphs, find all isomorphic pairs. If you find two graphs that are isomorphic, give the isomorphism from one to the other (either order is fine).



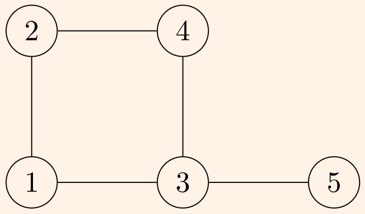
G_1



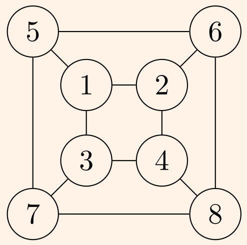
G_2



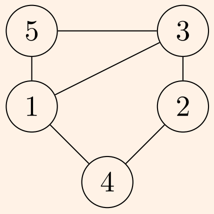
G_3



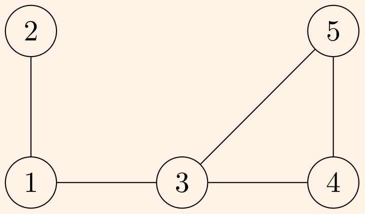
G_4



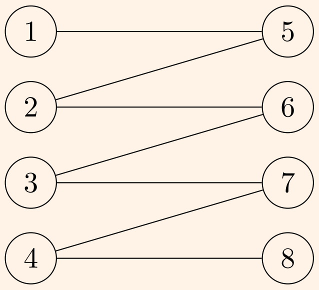
G_5



G_6



G_7



G_8

Question 1.6 (5 pts)**.** If we consider isomorphic graphs to be the same, how many distinct graphs are there with four vertices?

Question 1.7 (2 pts)**.** Given two graphs G_1, G_2, both with n vertices, how many bijections are there between the vertex sets of the two graphs?

Question 1.8 (5 pts)**.** Draw two isomorphic graphs with exactly 6 vertices and with exactly one isomorphism between them.

Let's now develop a deterministic procedure to check whether graphs are isomorphic to each other.

Question 1.9 (3 pts)**.** Suppose someone hands you two graphs (e.g. their vertex and edge sets) $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ which both have n vertices and m edges. Devise a method to check if the two graphs are isomorphic.

It might be that your method is not very fast to implement. Suppose I ask you to pay for your method (or *algorithm*).

Definition 1.3. The **cost** of an algorithm is defined as the sum of the costs of the basic operations required to implement the algorithm. For graphs, if you need to iterate through the elements of an edge set, it costs you $1 per element. If you want to check if an element is in a set, this also costs you $1. For instance, if I wanted to look at every edge in G_1, it would cost me \$$m$.

Question 1.10 (4 pts)**.** Compute the possible cost of your algorithm to Question 1.9 in the worst case. Your answer may depend on m and n. Don't worry about making the cost of the algorithm as small as possible; a correct algorithm and correct cost analysis is all we need.

For the next two questions, let C_{old} be the answer to Question 1.10.

Question 1.11 (2 pts)**.** If I double $m \mapsto 2m$, compute the new cost in terms of C_{old}.

Question 1.12 (2 pts)**.** If I increment the number of vertices $n \mapsto n+1$, determine an expression for the new cost in terms of C_{old}.

Thus, figuring out whether two graphs are isomorphic can be expensive. However, this doesn't necessarily mean that once you know the answer, *proving* to someone else that two graphs are isomorphic is expensive. Suppose there are two friends, Paula and Victor. Paula tells Victor that she thinks two graphs G_1 and G_2 are isomorphic. Victor seems skeptical, so Paula does the following to prove it.

System 1.4. For some graphs G_1 and G_2:

1. Paula hands Victor the complete evaluation table of a function, f, she claims is an isomorphism from G_1 to G_2. That is, she hands him a table with all possible inputs (the vertices of G_1) and their matching output:

vertex	f(vertex)
v_1	w_1
v_2	w_2
$\vdots$	$\vdots$
v_n	w_n

2. Victor checks if f is actually an isomorphism. If it is, he believes Paula's claim that the graphs are isomorphic. Otherwise, he doesn't believe Paula.

For all of our systems, if Paula sends any extraneous information that is not outlined in the scheme, Victor rejects automatically (here, she HAS to send an evaluation table of the correct size, even though it may be incorrect, e.g. if we have $G_1 \not\cong G_2$).

Question 1.13 (2 pts)**.** In terms of $n = |V_1|$ and $m = |E_1|$, compute the size of the evaluation table of the isomorphism f. Assume that a vertex is of size 1.

Question 1.14 (4 pts)**.** Write an algorithm, using the evaluation table that Paula provided, that checks Paula's claim.

Question 1.15 (3 pts)**.** Again, consider charging the algorithm with the cost scheme outlined in Definition 1.3. Compute the cost of your algorithm in the worst case. Any operation without an explicitly defined cost can be assumed to be free.

Thus, it is often much more efficient to check a proof than it is to solve a problem from scratch! This makes intuitive sense; checking is a "linear" operation, involving just checking that each step is sufficiently justified by previous steps. Presenting your own proof requires much more thinking and insight[1]. This motivates the following definition.

[1]This is exactly the distinction between the complexity classes $\mathcal{P}$ and $\mathcal{NP}$, the subjects of one of the Millenium Problems!

Definition 1.5. For the purposes of a graph problem with one or more graphs G_i, an **efficient** algorithm means that the cost of the algorithm is **a polynomial** in terms of $m = \max_i |E_i|$ and $n = \max_i |V_i|$ (the exponents cannot depend on m or n). Algorithms with costs $\$2^n$ and $\$m^n$ are not efficient, but $\$m^2n$ and $\$m^{100}$ are.

You can assume the following conjecture is true for the rest of the round (although no one has proved it yet):

Conjecture 1.6. There exists no efficient algorithm in general to determine whether two graphs G_1 and G_2 are isomorphic.

We suspect that even though it's easy to check an isomorphism, it's hard to find one.

2 Probabilistically-Checkable Interactive Proofs (31 pts)

In mathematics, a proof is generally accepted if the person reading it (a verifier) finds that it is logically consistent and justifies the claims made. However, this is not the only way to prove something. For instance, if a friend claimed to know the winning numbers to a lottery ahead of time and hit the jackpot 5 times in a row, it would generally be acceptable to believe that they have a means of knowing those numbers ahead of time, even if there is a chance that they had simply been lucky in guessing. By asking them to guess more and more lottery numbers, the chances get better and better.
Suppose a newly-released, efficient algorithm is claimed to simulate fair dice rolls. In this case, the intention is that each outcome of a die has a $\frac{1}{6}$ chance of occurring. Paula claims that the algorithm is faulty, and each roll will instead always produce the same number.

System 2.1. To prove the die-rolling algorithm returns the same result for each roll, Paula does the following.

1. Paula tells Victor what the algorithm will roll.
2. Victor uses the efficient algorithm to simulate the rolls of the die once.
3. If Victor's simulated roll of the dice matches what Paula predicted, he believes her claim that the algorithm produces the same result for each roll. Otherwise, he doesn't believe Paula since her prediction was wrong.

In this case, if the algorithm returns the same result for each roll, and Paula knows this result, she should always successfully predict the outcome of the simulated die roll. Consequently, Victor would always believe her.

Question 2.1 (2 pts)**.** Suppose the die-rolling algorithm correctly simulates a fair die, returning one of six random outcomes, each with probability $\frac{1}{6}$. In this case, Paula's claim would be false. Compute the probability that she still correctly predicts the outcome of the simulated roll, which would convince Victor that the die simulation is faulty.

Under the right conditions, systems like the one above are **probabilistically-checkable interactive proofs**.

Definition 2.2. A **probabilistically-checkable interactive proof** (PCIP) system is a coordinated algorithm between two players, named Victor and Paula. It consists of back-and-forth communication between the two parties, wherein Paula is trying to prove a statement x to Victor, and Victor can only run *efficient* algorithms.

- The **completeness** of the system is the probability that Victor believes x is true given that Paula's claim is actually true. In other words, it measures Paula's ability to prove a true statement to Victor.
- Suppose x is false, but Paula is trying to make Victor believe it is true. The **soundness** of the system is the maximum probability, over all possible strategies of Paula (where she has to send the same messages as if she were honest), that Victor believes Paula. In other words, it measures Victor's ability to avoid believing false statements from Paula.

We require that a PCIP satisfies the following properties:

1. The completeness is at least $2/3$.
2. The soundness is at most $1/3$.

For instance, the completeness of System 2.1 is 1, while the soundness of the system is the answer to Question 2.1.

System 2.3. Suppose that the dice algorithm from System 2.1 is faulty, but returns the number 3 with probability $\frac{1}{2}$ and the other numbers each $\frac{1}{10}$ of the time. Suppose Paula knows this and makes the claim to Victor that the algorithm has these new probabilities. Paula uses the same procedure as System 2.1, where she always tells Victor that a 3 will be rolled.

Question 2.2 (2 pts)**.** Compute the completeness of System 2.3; that is, when the distribution is indeed like this, find the probability that the system succeeds.

In fact, the $1/3$ and $2/3$ values in Definition 2.2 are somewhat arbitrary. Indeed, many other sets of values work, as long as the completeness is greater than the soundness.

Definition 2.4. For a PCIP system S, define the **repetition system** $\text{REP}_{\ell,T}(S)$ as the system where the pair repeats the system S ℓ times independently (i.e. all randomness between runs is independent) with a threshold $T \in [0, 1]$, where Victor believes Paula overall if he believes her claim after at least $T\ell$ of the repetitions.

Question 2.3 (5 pts)**.** Compute some T and ℓ such that $\text{REP}_{\ell,T}(\text{System 2.3})$ has completeness greater than $\frac{2}{3}$ and the soundness less than $\frac{1}{3}$.

For large enough ℓ, we can utilize the **law of large numbers**.

Theorem 2.5. The **law of large numbers** says that when a random experiment (such as a PCIP) is repeated enough times, the fraction of trials that correspond to each possible outcome gets arbitrarily close to the probability of that outcome happening.

For instance, suppose Victor has a probability p of believing Paula in a PCIP and a probability $1 - p$ of not believing her. If this PCIP is run for large enough ℓ, then Victor will believe Paula in about $p\ell$ of those trials and will not believe her in about $(1-p)\ell$ of those trials.

Question 2.4 (5 pts)**.** Suppose that Victor and Paula run a PCIP system S. Find an explicit threshold T where there exists an ℓ such that the soundness of $\text{REP}_{\ell,T}(S)$ is arbitrarily close to 0 and the completeness of $\text{REP}_{\ell,T}(S)$ is arbitrarily close to 1. Justify.

To illustrate another situation where a PCIP system could be useful, let's tackle a problem similar to Question 1.14. Paula and Victor still have access to two graphs G_1, G_2 with the same numbers of vertices and edges, but Paula wants to prove that the graphs are NOT isomorphic (this is her statement x).

Question 2.5 (3 pts)**.** Explain why our previous algorithm from Question 1.14, of sending the evaluation table of an isomorphism and checking it, is insufficient for proving that two graphs are NOT isomorphic.

Thus, we must turn to a PCIP system.

System 2.6. Consider the following system:

1. Victor selects at random one of the two graphs G_1 or G_2 and sends to Paula a random isomorphic copy of this graph, G'.
2. Upon receiving G', Paula tells Victor which of G_1 or G_2 she thinks G' was copied from (i.e. if she thinks it's G_b, then she sends the number $b \in \{1,2\}$).
3. If Paula tells Victor the correct answer, then Victor believes that G_1 and G_2 are not isomorphic; otherwise, he rejects Paula's proof.

Question 2.6 (4 pts)**.** State an efficient procedure to generate an isomorphic copy of a graph uniformly at random. Assume that you can generate a random number in $\{1,2,\dots,N\}$ for \$1. Give the cost of your procedure (still charging the set operations from before).

Question 2.7 (3 pts)**.** Suppose Paula wasn't honest and the graphs were actually isomorphic. Explain why Paula has no hope, past random guessing, of figuring out which graph G' came from.

However, as is, this isn't quite a PCIP since the completeness and soundness are lacking a bit.

Question 2.8 (2 pts)**.** Compute the completeness of this system. That is, if the graphs are not isomorphic and Paula is able to tell them apart, then compute the probability Victor believes this.

Question 2.9 (2 pts)**.** Compute the soundness of this system. That is, if the graphs are isomorphic and Paula is lying, then compute the maximum probability Victor believes her. *HINT: Paula sends exactly one piece of information to Victor and cannot send anything else.*

Question 2.10 (3 pts)**.** Explain how to amplify soundness and completeness to the $1/3$ and $2/3$ thresholds that are necessary, i.e. to make the resulting scheme a PCIP.

3 Zero-Knowledge Proofs (37 pts)

System 1.4 provides a way for Paula to prove to Victor that two graphs are isomorphic. However, it requires her to give an isomorphism f to Victor. In some situations, Paula may not necessarily want to give out full information during a proof. To determine what Victor learns from running an interactive proof with Paula, we consider the **transcript of communication** of the proof.

Definition 3.1. A **transcript of communication** is a record of the messages that were exchanged between Victor and Paula.

For instance, if Victor and Paula run System 1.4, the transcript would be the entire evaluation table of f that Paula passes to Victor. Other information during this proof that wasn't shared, such as Victor's work to verify Paula's isomorphism, is not included in the transcript. In other words, the transcript of communication should only include information that is visible to both Victor and Paula during the proof.

Question 3.1 (2 pts)**.** Describe what information is included on the transcript of communication from running System 2.6.

Question 3.2 (4 pts)**.** Suppose an *efficient* algorithm M exists that produces the transcript of communication for a proof system. Explain intuitively why the existence of M indicates that Victor learns no secret knowledge from interacting with Paula within the proof system. *HINT: Victor can run efficient algorithms.*

Many proof systems, such as System 2.6, rely on randomness that occurs during the proof. In these cases, even when using the same graphs, the transcript of communication between Victor and Paula is not fixed due to the randomness that occurs. In this case, we consider all possible transcripts that could occur as well as the probability each one occurs. A function that describes this relation is a **probability distribution**.

Definition 3.2. In general, a **probability distribution** for a random experiment is a function $\mathbb{P}$ that takes in any possible outcome as an input and outputs the probability that outcome occurs.

For example, the possible outcomes of a roll of a six-sided die are 1, 2, 3, 4, 5, and 6. Since the die is fair, for any integer $1 \leq x \leq 6$, $\mathbb{P}(x) = \frac{1}{6}$.
Two probability distributions are **equal in distribution** if, for each outcome in the first distribution, a corresponding outcome in the second distribution also has the same probability. For instance, the distribution of whether the roll of a fair six-sided die is even or odd and the distribution of the result of flipping a fair coin are equal, since the probability of each event (even or odd, heads or tails) is $\frac{1}{2}$.

Question 3.3 (3 pts)**.** Give another example of two simple random experiments whose outcomes are equal in distribution, but the outcomes are not necessarily the same.

Since a transcript of communication contains all messages Victor receives from Paula during a proof, we can use this transcript to determine whether or not Victor learned any unnecessary information during the proof. Proofs where Victor does not learn any additional information are said to be **zero-knowledge**.

Definition 3.3. A **zero-knowledge proof** (ZKP) system to prove a statement x is a PCIP system and an *efficient* algorithm M (called the **simulator**) where the output generated by M on input x is equal in distribution to Paula and Victor's transcript of communication (in the case where the statement is correct and Paula knows the proof). That is, if T is some transcript of messages, we must have $\mathbb{P}(M(x) = T) = \mathbb{P}(\text{Paula and Victor make } T)$.

In a sense, M "indistinguishably simulates" a possible communication between Victor and Paula.[2] We can reanalyze our previous systems and see if they have the zero-knowledge property.

Question 3.4 (6 pts)**.** Show that the algorithm you designed in Question 1.14 is a PCIP system, but not a ZKP system. *HINT: You may assume conjecture 1.6, that there is no efficient way to tell if two graphs are isomorphic.*

Question 3.5 (4 pts)**.** Show that System 2.6 is a ZKP system.

Now, let's refine the scheme from Question 1.14 to make it zero-knowledge. We will do this by introducing some randomness. Suppose we have two graphs G_1, G_2 that Paula wants to prove are isomorphic.

System 3.4. Consider the following system, where Paula knows an isomorphism f from G_1 to G_2. Assume that $|V_1| = |V_2|$.

1. Paula chooses a random bijection g and sends $H = g(G_2)$, the graph you get by putting G_2 through this isomorphism.
2. Victor chooses a random number $b \in \{1, 2\}$ and sends b to Paula.
3. Paula then sends the evaluation table of a bijection h from the vertices of G_b to the vertices of H. If $b = 1$, $h(v) = g(f(v))$. If $b = 2$, $h(v) = g(v)$.
4. Victor believes Paula if h is an isomorphism (it respects edges).

Let's analyze this scheme.

Question 3.6 (4 pts)**.** Compute the soundness of this scheme.

Question 3.7 (2 pts)**.** Compute the completeness of this scheme.

Question 3.8 (8 pts)**.** Note that the transcript of messages sent is the triple ($H, b,$ evaluation table of h). Find an efficient algorithm M to generate the transcript between the two players. Analyze the cost of your algorithm to show it is efficient.

Thus, using the completeness/soundness amplification we noted prior, System 3.4 is a ZKP system.

Now that we've found examples and non-examples of ZKP systems, let's consider how they behave when used together. For instance, suppose Victor wanted to solve question Question 1.5 with help from Paula. System 2.6 only allows Victor to ask about 2 graphs at a time, but he has to check 8. To resolve this, he could just run the system multiple times, once with each combination of 2 different graphs.

Definition 3.5. Given ℓ PCIP Systems $S_1, S_2, \ldots, S_\ell$, their **serial composition** is the result of running them one after another independently. That is, Paula tries to convince Victor a statement x_1 is true through a PCIP protocol S_1, then convinces Victor about a (possibly different) statement x_2 through S_2, and so on. All messages related to the system S_j must be sent/received before the first message of S_k, for all $j < k$. You may assume that l is independent of the time it takes to run each PCIP system.

Question 3.9 (4 pts)**.** Show that the serial composition of multiple zero-knowledge proofs will always result in an interaction with the zero-knowledge property.

[2]Technically, this is the notion of honest-verifier perfect zero-knowledge, but that distinction does not matter for us here.

4 ZKP Systems From Other Hardness (32 pts)

In computer science, we often struggle to find efficient algorithms for problems, so we conjecture that they are hard. As we saw with graph isomorphisms, assuming this allows us to get zero-knowledge schemes. Let's use another common conjecture to form another zero-knowledge proof scheme. For the purposes of this section, p is a very large prime number.

Definition 4.1. An integer $g \pmod p$ is called a **generator** if every number in $\{1, 2, \ldots, p-1\}$ can be written as $g^a \pmod p$ for some a.

Here, in the modular arithmetic setting, we charge costs a little differently: adding or multiplying two numbers $\bmod p$ costs \$1. Making random numbers still costs the same. An algorithm is **efficient** in modular arithmetic if it's a polynomial in $\log_2 p$, the number of binary digits in p.

Question 4.1 (6 pts)**.** Devise an efficient algorithm for computing $g^a \bmod p$.

It turns out that undoing the operation is much more difficult.

Conjecture 4.2 (Discrete Logarithm Assumption)**.** Given a generator g and $g^a \mod p$ for some a in $\{1, 2, \ldots, p-1\}$ that you do not know, there is no expected efficient algorithm to find a (i.e. one whose cost is, on average over all randomness, polynomial in $\log_2 p$).

Before we can harness this, we should see why modular arithmetic plays nicely with randomness.

Question 4.2 (3 pts)**.** Show that given a fixed number N, then if we randomly pick R in $\{0, 1, \ldots, p-1\}$, then $N + R \bmod p$ is equal in distribution to R.

Question 4.3 (3 pts)**.** Show that given a fixed number N not equivalent to $0 \pmod p$, if we randomly pick R in $\{1, 2, \ldots, p-1\}$, then $NR \bmod p$ is equal in distribution to R.

Suppose now that everyone has access to the same prime p and a generator $g \pmod p$. Paula picks a random number α from the set $\{1, 2, \ldots, p-1\}$ and gives Victor $u = g^\alpha$. Victor and Paula want to make a scheme so Victor can identify Paula in communications, without Victor himself being able to impersonate Paula.

Question 4.4 (20 pts)**.** Construct a ZKP scheme that allows Paula to convince Victor that she knows α, and does not allow others to convince Victor they know α. Here is a possible scheme with some steps removed that you can use as a template.

1. When she wants to log in, Paula chooses randomly $\alpha_t \in \mathbb{Z}_p$ (where $\mathbb{Z}_p$ is the set $\{0, 1, 2, \ldots, p-1\}$), computes $u_t =$ ______ $\bmod p$ and sends u_t to Victor.
2. Victor chooses randomly $c \in \mathbb{Z}_p$ and sends c to Paula.
3. Paula computes $\alpha_z =$ ______ $\bmod p$ and sends it to Victor.
4. Victor accepts the proof if $g^{\alpha_z} \equiv$ ______ $\pmod p$.

Discuss the soundness and completeness of your scheme, and provide an efficient simulator for the transcript.