1 Short Answer I (10 questions) - 20 points

For each of the problems circle *true* if the statement is *always* true, circle *false* otherwise. There is no partial credit for these questions.

(a) If a problem is in NP, then it must also be NP-hard.

Solution: True (False)

Nope. Integer factorization is an example of a problem in NP that is not NP-hard (also all of P as far as we know).

■

(b) There exists a polynomial time reduction from every NP-hard problem to every NP-complete problem.

Solution: True (False)

Nope. The Halting problem is NP-hard but you can't reduce it to SAT (or any other computable problem). ■

(c) If $A \leq_P B$ and B is NP-Complete, then A is NP-Complete.

Solution: True (False)

Nope. Remember the reduction direction $X \leq_P NP$ simply means that X is in NP (the right side is a upper bound). ■

(d) There exists a polynomial time reduction from every problem in NP to every problem in P.

Solution: True (False)

Nope. This would imply P=NP. ■

(e) All recursively enumerable problems are NP-hard.

Solution: True (False)

Nope. All problems in P are recursively enumerable and yet they're not NP-hard. ■

(f) If a problem is NP-hard and co-NP-hard, then it must be in NP.

Solution: True (False)

Nope. The Halting Problem is in both but not in NP because it isn't computable. ■

(g) If a language is not recursively enumerable then its complement must be recursively enumerable.

Solution: True False

Not necessarily. It is true that if a language is recursively enumerable then its complement can't be. But that doesn't mean there aren't languages who themselves and their complement are non-recursively enumerable. Link to more info ■

(h) All languages that are decidable are solvable in polynomial space.

Solution: True False

That would imply all languages are solvable by a LBA, i.e. all languages are context sensitive. Nope. ■

(i) Determining if the language represented by a context-free grammar accepts all strings is decidable.

Solution: True False

We talked about this in lecture 24 (MT3 Review, the one right before the midterm). determining if a context free grammar accepts all strings is undecidable. ■

(j) Ignoring $\emptyset$ and Σ^* [1], even if P=NP, there can still be some problems that are in P/NP, but not are NP-hard.

Solution: True False

If P/NP, then all NP problems are in P, which means they have a polynomial time algorithm that solves them, which means that all the problems in NP are by definition polynomial time reducible to eachother, which means that SAt (or any other NP-hard problem) is reducible in polynomial time to every other NP problem. Ergo, there are no problems in NP that are not NP-hard if P=NP. ■

[1] I had to put this qualifier because technically you can't do reductions with trivial languages because reductions map yes to yes and no to no and those trivial sets either have only yes's or only no's.

2 Short Answer II (5 questions) - 10 points

Consider the "**NoTwoOnes**" problem defined by language L.

1. Let L be the language of strings that do not contain the sub-string **11**.
2. Let D be a decider for L (it accepts strings that do not contain the sub-string **11**).

We reduce from *3SAT* to the *NoTwoOnes* problem. A decider $S(x)$ for the 3SAT problem is constructed as follows, where ϕ is a 3-CNF formula.

```
S(φ):

   # Encode a Turing machine M_x as follows.
      M_x(w):
         for all possible variable assignments (x):
            if φ(x) == True and w does not contain 11
               return True
         return False

   return D(⟨M_x⟩)
```

(a) Circle all complexity classes that L belongs.

Solution: (P) (NP) (Decidable) Undecidable

■

(b) Is the reduction from **3SAT** to the **NoTwoOnes** correct? If so, prove the forward and backwards directions. If not, explain where the error is.

Solution: L is the language of strings that do not contain the substring 11. That is: $L = \{w \mid w \text{ does not contain } 11\}$.

Let D be a decider for L. The decider $D(w)$ works as follows:

- Returns **False** if the string w contains 11.
- Returns **True** otherwise.

- We are passing $\langle M_x \rangle$ into D. Thus, D is deciding whether the string encoding of M_x contains 11.
- The encoding $\langle M_x \rangle$:
  - Either contains 11 or does not contain 11.
  - Regardless, the 3-SAT solver is only determining whether the string encoding of the Turing machine M_x contains 11, **not if ϕ is satisfiable**.

■

(c) Does this reduction show that **NoTwoOnes** problem is NP-hard? Why or why not?

Solution: As shown above, the reduction from 3SAT to NoTwoOnes is incorrect. The 3SAT decider $S(\phi)$ always returns the same boolean value. Both **yes** and **no** instances of 3SAT map to the same output boolean value. Therefore, the reduction is invalid, and this does not demonstrate that NoTwoOnes is NP-Hard. ■

(d) Is the language $L(M_x)$ Decidable or Undecidable?

Solution: (Decidable) Undecidable

A language is **decidable** if there exists a Turing machine (TM) that can solve the problem. In this case, we have already provided pseudocode for M_x. Therefore, the language $L(M_x)$ is decidable. ■

(e) Is the reduction from **3SAT** to the **NoTwoOnes** problem polynomial in time?

Solution: The Turing machine M_x is never executed during the reduction, so the amount of work done by M_x is irrelevant—even if it is exponential. The runtime of the reduction is solely determined by the time required to encode M_x. Since encoding a Turing machine with a fixed number of lines takes $O(1)$ time, the reduction operates in polynomial time. ■

3 Classification I (P/NP) - 14 points

Is the following problem in P, NP, or some combinations of complexity classes? For each of the following problems, circle all the complexity classes that problem belongs to. Whatever class it is in, prove it!

A **LostMyPhone cycle** asks given a *directed* graph G, does G contain a path that first visits all n vertices exactly once and then visits those vertices in the reverse order. So if the graph has 3 vertices and is fully connected, a LostMyPhone cycle might look like $n_1 \to n_2 \to n_3 \to n_3 \to n_2 \to n_1$.

- INPUT: A directed graph G.
- OUTPUT: TRUE if there exists a path that first visits all n vertices exactly once, and then visits the vertices in the reverse order. FALSE otherwise.

Which of the following complexity classes does this problem belong to? Circle ***all*** that apply:

Solution: P (NP) (NP-hard) (NP-complete)

- **Proof of NPness**

  Certificate : A LostMyPhone(LMP) cycle C

  Certifier : Iterate over C and check if it is a valid LMP cycle

- **Proof of NP-hardness**
  Reduce from undirected Hamiltonian path(UHP) problem. For an arbitrary instance $G = (V, E)$ of UHP, construct a directed graph $G' = (V', E')$ where $V' = V$ and $E' = \{(u, v), (v, u) \mid (u, v) \in E\}$. That is, for each undirected edge in G, add two directed edges between the two nodes in both directions. Then, there exists a UHP in G if and only if there exists a LMP cycle in G'.

  ($\to$) Suppose there exists a UHP $(v_1, v_2, ..., v_n)$ in G.
  By construction, $(v_1, v_2, ..., v_n, v_{n-1}, ..., v_1)$ is an LMP cycle in G'.

  ($\leftarrow$) Suppose there exists an LMP cycle $(v_1, ..., v_{n-1}, v_n, v_{n-1}, ..., v_1)$ in G'.
  By construction, $(v_1, v_2, ..., v_n)$ is a UHP in G.

- **Proof of NP-completeness**
  Since LMP cycle is both NP and NP-hard, LMP cycle is NP-complete.

Grading

- **Complexity class**
  One point deducted per incorrect selection. Each of selecting P and not selecting NP counts as an incorrect selection.
- **Certificate and certifier**
  The certificate and the certifer must be clearly recognizable from the solution in order to get full credit. Partial credit(1.5) may be granted if the solution somewhat demonstrates your understanding of the two concepts.

- **Reduction**
  You must provide a valid reduction from a known NP-hard problem to LMP cycle. That is,

  (1) It must be possible to apply the reduction to any arbitrary instance of an NP-hard problem without ambiguity

  (2) The reduced instance must be a directed graph.

  The reduction is not valid if it involves finding specific nodes such as the beginning of a Hamiltonian path(HP), the endpoint of an HP, etc., since this requires solving the HP problem in advance while your goal is to construct an algorithm for HP. For this item, the correctness of the reduction is ignored - you get full credit as long as you provide a valid reduction. Partial credit(1.5) may be granted if the solution somewhat demonstrates the understanding of the concept of reduction.

- **Correctness of reduction**
  This item is graded based on the correctness of the reduction and your proof of correctness. For full credit,

  (1) Your reduction must be correct, so the reduced instance is a YES instance if and only if the original instance is a YES instance.

  (2) You must provide a valid bidirectional reasoning about the correctness of your reduction.

  Reduction from directed Hamiltonian path(DHP) by adding edges in the opposite directions is not a correct reduction since there exists a NO instance of DHP that is reduced to a YES instance of LMP cycle. You must briefly explain why the reduction is correct for full credit. Stating that YES instances are mapped to YES instances and NO instances are mapped to NO instances without explaning the reason won't suffice. Patial credit(2.5) may be granted if the reduction works for YES instances. You may get partial credit(1) for trying to reason bidirectionally, even though it is incorrect.

■

4 Classification II (P/NP) - 14 points

Is the following problem in P, NP, or some combinations of complexity classes? For each of the following problems, circle all the complexity classes that problem belongs to. Whatever class it is in, prove it!

The MostofSAT (**MostofSAT**) problem asks whether a conjunctive normal form formula ϕ has a truth assignment that satisfies the majority ($\geq m/2$) of the clauses.

- INPUT: A SAT formula ϕ.
- OUTPUT: TRUE if there exists a truth assignment that satisfies the majority of the SAT clause in ϕ. FALSE otherwise.

Which of the following complexity classes does this problem belong to? Circle ***all*** that apply:

Solution: (P) (NP) NP-hard Undecidable

The bold text indicates the minimum word needed to present to get full credit.

We claim that for any SAT formula ϕ, there is always a truth assignment that satisfies the majority of the SAT clause ϕ. Assume n is the number of variables and m is the number of clauses, pick $A = (0,1)^n$ which is a random assignment for those n variables. Suppose the assignment A satisfies k clauses in ϕ. If $k \geq \frac{m}{2}$, we know A is an assignment that satisfy the MostofSAT and thus we should return true. If $k < \frac{m}{2}$, then we observe the complement of A, denoted as $\bar{A}$. For every clauses c in those $m-k$ clauses that evaluates to False under A, we know that all variables in c evaluates to False under A. By setting $\bar{A}$ as our assignment, for every clauses c in those $m-k$ clauses, at least one variable will evaluate to True, making clause c evaluate True. Thus, under assignment $\bar{A}$, we have at least $m-k > m-\frac{m}{2} \geq \frac{m}{2}$ clauses evaluates to true. So, we know for any assignment A, either A or $\bar{A}$ will satisfy the majority of the clauses.

We always return true, this takes $O(1)$ time, and thus the algorithm runs in P.

The P algorithm that solves the problem is not unique, but here we only present the fastest algorithm.

Since the problem is in P, it's also in NP.

One can also prove NP by using a certifier and a certificate. The Certificate is A, an assignment for n variables. The Certifier checks by counting how many clauses does A satisfy, and compare the counter with $\frac{m}{2}$. The counting takes $O(m)$ time, which is polynomial. Therefore, MostofSAT is in NP

List of Common mistakes:

1. Reduction
   (a) If X is a known NPC problem, $X \leq_P Y$ only implies $Y \in NPH$, it does not show $Y \in NP$ even if the reduction is completely correct.
   (b) $Mostof SAT \leq_P SAT$ gives no useful information, as $Mostof SAT$ could either be in P or NPH.
2. NP
   (a) A certifier and a certificate must be presented to get full credit, simply mentioning

the definition of NP will only receive partial credit, as you are proving why the $MostofSAT \in NP$, you can't answer the question with the question.

(b) For a question to be in NP, a certifier that runs in **polynomial** time must be shown.

3. P

(a) Even we don't require the fastest algorithm for showing P, the algorithm presented still has to run in P time. An algorithm is said to be of polynomial time if its running time is upper bounded by a polynomial expression in the size of the input for the algorithm, which is equivalent for saying $F(n) = O(n^c)$ for some positive constant c that is independent from the input. Thus, these runtime does not count as P: $O(n^{m/2}), O(m^{m/2}), O(2^{m/2})$. Algorithm that using brute force to try every possible alignment runs in $O(2^n)$ as there are $O(2^n)$ different assignments.

(b) If the submission claimed $MostofSAT \in P$, but the algorithm is wrong not does not run in P time, it receives no points for claiming that $MostofSAT \in NP$ as $P \subset NP$

4. SAT

(a) The input to $MostofSAT$ is a SAT clause ϕ, not a specific assignment. You can't tell whether a clause evaluates to True or False without sampling a specific assignment.

(b) There is no clause that always evaluates to False, as a clause contains only variables.

■

5 Classification I (Decidability) - 14 points

Are the following languages decidable?

$$\text{ACCEPT4STRINGS}_{TM} = \{\langle M \rangle \mid M \text{ is a } \textbf{\textit{TM}} \text{ and accepts exactly 4 strings. } (|L(M)| = 4)\}$$

Solution: Decidable (Undecidable)

We give the following reduction from the Halt decider. Suppose there is an algorithm DECIDEFOUR that correctly decides the language of all $\langle M \rangle$ that accept exactly 4 strings. Then we can solve the halting problem as follows:

```
DECIDEHALT(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        run M on input w
        if x ∈ {0, 1, 10, 01}
            return TRUE
        else
            return FALSE
  if DECIDEFOUR(⟨M′⟩)
      return TRUE
  else
      return FALSE
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w.

Then M' accepts *every* input string x.

In particular, M' accepts the 4 strings $\{0, 1, 10, 01\}$.

So DECIDEACCEPTFOUR accepts the encoding $\langle M' \rangle$.

So DECIDEHALT correctly accepts the encoding $\langle M, w \rangle$.

$\Longleftarrow$ Suppose M does not halt on input w.

Then M' diverges on *every* input string x.

M' does not accept any strings, and in particular does not accept 4 strings.

So DECIDEACCEPTFOUR rejects the encoding $\langle M' \rangle$.

So DECIDEHALT correctly rejects the encoding $\langle M, w \rangle$.

To get full points, a solution must select "Undecidable", attempt to solve using a reduction from an undecidable language (such as the Halting Problem), and give a correct proof of reduction.

■

6 Classification II (Decidability) - 14 points

Are the following languages decidable? For each of the following languages,

- Circle one of "decidable" or "undecidable" to indicate your choice.
- If you choose "decidable", show your choice correct by describing an algorithm that decides that language. If you choose "undecidable", show your choice correct by giving a reduction proving its correctness.
- Regardless of your choice, explain *briefly* (i.e., in few sentences, diagrams, *clear* pseudo-code) why your choice is valid.

$$\textsc{AcceptExpSpace}_{TM} = \{\langle M, w\rangle \mid M \text{ is a } \textbf{\textit{TM}} \text{ and accepts } w \text{ in } 2^{|w|} \text{ space. }\}$$

Solution: (Decidable) Undecidable

Please refer to HW 9 - P3(c)

We can construct a Turing machine M' to decide $\textsc{AcceptExpSpace}_{TM}$ as follows. Suppose M' has $\langle M, w\rangle$ as its input. We assume M has the states Q and tape alphabet Γ. M' runs M on w for $k \triangleq |Q| \times 2^{|w|} \times |\Gamma|^{2^{|w|}}$ steps.

- $|Q|$ is number of states of M
- $2^{|w|}$ is number of possible tape head positions
- $|\Gamma|^{2^{|w|}}$ is the maximum number of possible strings that can be on the first $2^{|w|}$ cells

Thus, k is an *upper bound* on the number of possible configurations of M if M only ever accesses the first $2^{|w|}$ cells. Note that if M runs for more than k steps while using only $2^{|w|}$ space:

- M must revisit a previous configuration because there are only k distinct configurations
- If M revisits a configuration, it enters a cycle and cannot make progress
- This means M will loop indefinitely and never accept w after revisiting a configuration

This implies that if M doesn't accept w in k steps while accessing only the first $2^{|w|}$ cells, M would *never* accept input w after accessing only the first $2^{|w|}$ cells.

If M accepts w in k steps *while accessing only the first* $2^{|w|}$ *cells on its tape*, M' accepts $\langle M, w\rangle$. Otherwise, M' rejects $\langle M, w\rangle$. M' decides $\textsc{AcceptExpSpace}_{TM}$ and so $\textsc{AcceptExpSpace}_{TM}$ is decidable.

■

7 Classification (Mixed) - 14 points

Same type of problem as before, except now you have to choose between a mix of computational and algorithmic complexity classes.

Given the language:

$$\text{HOLIDAY}_{TM} = \{\langle M \rangle \mid M \text{ rejects only on the string "Told you so" }\}$$

Which of the following classes does this language belong to? Whatever you choose, *succinctly* prove it!

Solution: P NP (NP-hard) NP-complete Decidable (Undecidable)

NP-Hardness Proof

- Assuming we have the black box HOLIDAYSOLVER, we reduce (polynomial time) 3SAT to HOLIDAY as below:

```
3SATSOLVER(φ):
  Encode the following Turing machine M:
      M(x):
        For all 2^n assignments of 0/1 to variables in φ:
            If φ evaluates to true under the current assignment:
                If x == "Told you so",:
                    return False.
        return True
  if HOLIDAYSOLVER(⟨M⟩)
      return TRUE
  else
      return FALSE
```

- If ϕ is satisfiable:
  - M encodes a TM that only rejects "Told you so."
  - Therefore, HOLIDAYSOLVER returns True.
- If ϕ is not satisfiable:
  - M encodes a TM that always returns True.
  - M is not a TM that only rejects "Told you so."
  - Therefore, HOLIDAYSOLVER returns False.

As we have showed that 3SAT is solvable by HOLIDAY, and thus HOLIDAY is proven NP-hard.

Decidability Proof

- Assuming we have the decider DECIDEHOLIDAY, we reduce (polynomial time) HALTONINPUT to HOLIDAY as below:

```
HaltOnInput(⟨M′, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        Run w on M
        return x ≠ "Told you so"
  if DecideHoliday(⟨M′⟩)
      return True
  else
      return False
```

- If M halts on w:
  - M' is a TM that only rejects "Told you so."
  - Holiday($\langle M' \rangle$) returns True.
- If M hangs on w:
  - M' is not a TM that rejects only "Told you so."
  - Holiday($\langle M' \rangle$) returns False.

So we have showed that Halt is decidable if we have the decider for HOLIDAY, and as halt is known to be undecidable, HOLIDAY is proven undecidable.

Some grading criteria:

- Just stating "We need to reduce A to B" without reduction detail counts as insufficient.
- Any reduction not relating the string input "Told you so" is counted wrong as there's no connection build between problems.

■

EXTRA CREDIT (1 pt)

What does NP stand for?

Solution: Non-deterministic polynomial-time ■