CISC-365*
Test #1
January 30, 2019

Student Number (Required) ______________________

Name (Optional)________________________________

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, **but will not be re-marked under any circumstances.**

The test will be marked out of 50.

Question 1	/15
Question 2	/10
Question 3	/10
Question 4	/15
TOTAL	/50

QUESTION 1 (15 Marks)

Let X be a problem in the NP class. The details of X are unimportant but you can assume that each instance of X consists of a set of n integers, and another integer t.

(Parts (a) through (e) are independent of each other. Each part is worth 3 marks)

(a) Suppose we find an algorithm that solves X in $O(2^n)$ time. Does this give us any information about whether X is in P, or whether X is NP-Complete? Explain.

Solution: This gives us no information. Problems in NP can all be answered in exponential time by examining all possible solutions.

(b) Suppose we are able to prove that every possible algorithm for X requires at least 2^n steps. Does this give us any information about the classes P, NP, and NP-Complete? Explain.

Solution: This would prove that P != NP because now we know there is at least one problem in NP that is not in P. It would prove that no NP-Complete problem can be solved in polynomial time (even if X itself is not NP-Complete).

(c) Suppose we are able to show that $X \propto k\text{-}Clique$. Does this give us any information about whether X is in P, or whether X is NP-Complete? Explain.

Solution: This gives us no information. We know X is in NP, and we know k-Clique is NP-Complete. From these facts we already know that X $\propto$ k-Clique

(d) Suppose we are able to show that $k\text{-}Clique \propto X$. Does this give us any information about whether X is in P, or whether X is NP-Complete? Explain.

Solution: We now know X is NP-Complete because a known NP-Complete problem reduces to X. Based on this knowledge we are very confident that X is not in P

(e) Suppose we find an algorithm that solves X in $O(n^t)$ time (remember that t is part of the instance definition). Does this give us any information about whether X is in P, or whether X is NP-Complete? Explain.

Solution: this gives us no information. $O(n^t)$ cannot be classed as polynomial time because t is not fixed. We have no evidence that X is NP-Complete.

Marking:

For each part:

Correct answer and reasonable explanation	**3/3**
Correct answer and poor or no explanation	**2/3**
Incorrect answer with some explanation	**1/3**
Incorrect answer with no explanation	**0/3**

QUESTION 2 (10 Marks)

The 3-Colouring Problem 3COL: Given a graph G on n vertices, can we colour the vertices of G using no more than 3 colours in such a way that no vertices that are joined by an edge have the same colour?

The 2-Colouring Problem 2COL: Given a graph G on n vertices, can we colour the vertices of G using no more than 2 colours in such a way that no vertices that are joined by an edge have the same colour?

3COL is known to be NP-Complete. However there is a polynomial-time algorithm for 2COL. We can call this algorithm 2C-ALG.

Consider this algorithm for 3COL:

```
# Let the colours be red, yellow, blue
For each subset T of the vertex set of G: {
     if T contains any vertices that are adjacent:
          skip this T
     else:
          colour all vertices in T red
          temporarily delete these vertices from G
          use the polynomial-time 2C-ALG algorithm to see if
               the remaining vertices can be properly
               coloured with yellow and blue
          if the answer is "Yes": print "Yes" and exit
          else: restore G to its original state
}
print "No"     # all attempts to 3-colour G have failed
```

This algorithm correctly solves 3COL .

Does this algorithm prove P = NP? Explain why or why not. If this space is too small for your answer, please use the back of this page.

Solution: The algorithm does not prove P = NP. Each iteration of the "for each" loop executes in polynomial time, but there are 2^n subsets of the vertex set of G so the loop may execute 2^n times. Thus the complexity of this algorithm is not polynomial.

Marking:

Any solution that recognizes that there are 2^n subsets to be checked, so the algorithm is not polynomial	**10/10**
Any solution that says the algorithm takes exponential time without relating it to the number of subsets of the vertex set	**7/10**
Any solution that says the algorithm does not prove P = NP but gives an invalid explanation, such as "These problems are not in NP"	**5/10**
Any solution that says the algorithm does not prove P = NP but gives no reason	**4/10**
Any solution that says the algorithm does prove P = NP, and tries to justify it	**2/10**
Any solution that says the algorithm does prove P = NP, with no explanation	**1/10**

QUESTION 3 (10 marks)

Consider this variant of the Subset Sum problem:

25_Value_Subset_Sum: Given a set S of exactly 25 integers and a target integer k, does S contain a subset that sums to k?

Prove this problem is in **P** by describing an algorithm to solve any instance of the problem in polynomial time. You are not required to express your algorithm in a programming language – simply explain it in sufficient detail to demonstrate that it runs in polynomial time. You do not need to compute the exact order of your algorithm.

Solution: S has exactly 2^{25} subsets, which is a large but constant number. Therefore we can examine all subsets of S in constant, ie O(1) time.

Marking:

Any solution that correctly explains that the problem can be solved in O(1) (ie constant) time	10/10
Any solution that proposes an algorithm that runs in $O(n^k)$ time for some k > 1	7/10
A solution that proposes an algorithm that actually runs in exponential time	4/10
A solution that proposes an algorithm that does not solve the problem	1/10

QUESTION 4 (15 Marks)

Recall the *Partition Problem*: Given a set of integers $S = \{a_1, a_2, ..., a_n\}$, does S contain a subset that sums to exactly $\frac{\sum_i a_i}{2}$ (ie, half of the total sum) ?

We know that Partition is NP-Complete.

Consider this problem:

P_A : Given a set of integers T (which may contain duplicate values), can T be divided into 3 disjoint subsets that all sum to the same value?

For example, if $T = \{1, 1, 1, 3, 4, 5, 5, 7, 9\}$ then the answer to P_A is "Yes" because T can be divided into $\{1, 1, 1, 4, 5\}, \{5, 7\}, \{3, 9\}$ each of which sums to 12.

(a) [5 marks] Prove that P_A is in the class NP

Solution: P_A is clearly a decision problem. Let T be any instance of P_A with n elements. If the answer is "Yes" and we are given the three subsets, we can sum each of the subsets in O(n) time, and confirm that the sums are equal in O(1) time. Therefore the "Yes" solution can be verified in polynomial time, so P_A is in NP

(b) [10 marks] Prove that $Partition \propto P_A$

Solution: Let $S = \{a_1, a_2, \ldots, a_n\}$ be an instance of Partition. Construct an instance T of P_A as follows:

Compute $x = \sum_i a_i$

If x is odd,

$T = \{1\}$

If x is even,

let $y = \frac{x}{2}$

$T = S \cup \{y\}$

This transformation clearly takes O(n) time.

Proof that the transformation is answer-preserving:

Suppose the answer to the Partition Problem on S is "Yes"
Then S can be divided into two subsets that each sum to $\frac{\sum a_i}{2}$, ie they each sum to y. Let these subsets be S_1 and S_2. Then T can be divided into S_1, S_2 and $\{y\}$, each of which sums to y – so the answer to P_A on T is "Yes"

Now suppose the answer to P_A on T is "Yes". We know T cannot be {1}, so we know S has an even sum. The sum of all elements of T is $3 * y$, so each of the three subsets with equal sum must sum to y. The added value y must be in one of the three subsets, and it must be alone in that subset. Thus the other two sets each sum to y (which equals $\frac{x}{2}$), and they form a partition of S. Thus the answer to Partition on S is "Yes".

Thus the transformation is answer-preserving.

Marking:

Part (a): Essential points:

Decision problem	1 mark
Yes answers verifiable	2 marks
Verification in polynomial time	2 marks

Part (b):

Polynomial time transformation	3 marks
Answer-preservation	
Correct proof	7 marks
Incorrect or incomplete proof	3 marks
Claim without proof	1 mark

Note that the transformation needs to deal with all possible instances of Partition. My answer separates out sets with an odd total sum – student answers may deal with this differently but it must be true that the constructed instances of T contain only integers.