What exactly is a hard problem?

1. What do we mean when we say that a decision problem is tractable? What is the complexity class **P** [4]

Solution:
Tractable - It can be solved by an algorithm of time complexity $O(n^k)$; that is, polynomial time
P - The complexity class of efficiently solvable problems

2. Give two objections to the definition of a tractable problem as a problem solvable in $O(n^k)$ time [4]

Solution:

- What about the hidden constant
- What if the k is massive

3. What do we mean when we say that a decision problem is efficiently checkable? [3]

Solution:
Given a potential witness that some instance is a yes-instance, we can check in polynomial time whether the witness is indeed a witness

4. What is a non-deterministic algorithm? How does a non-deterministic algorithm accept some input? How do we measure time taken by some non-deterministic algorithm on some input [6]

Solution:
It is an algorithm which guesses the solution, then check if it is correct.
It accepts an input when tested for correctness it passes
One "thread" is executed for every possible guess, acceptance is where there is at least one accepting thread.

Time taken is the time taken by the shortest accepting computational thread

5. What is the complexity class **NP** [2]

Solution:
The complexity class of decision problems that are efficiently checkable

6. Explain why the decision problem whose instances are pairs (G,b) with G a graph and b a natural number, and whose yes-instances are instances where G can be properly b-coloured is in **NP** [3]

Solution:
Looking at each node and seeing if any of its neighbours are the same colour can be calculated in polynomial time

7. What do we mean when we write **P** ⊆ **NP**? Explain why **P** ⊆ **NP** [3]

Solution:
We could try every possible combination and check it, as in a non deterministic algorithm

8. What is the **P** versus **NP** problem [2]

Solution:
Are all problems in NP also in P

9. Define precisely the notion of a polynomial-time transformation [4]

Solution:
A polynomial time transformation from X to Y is an algorithm α that:

- Takes an instance I of X as input and provides an instance $\alpha(I)$ of Y as output
- Is such that an instance I of X is a yes-instance iff the instance $\alpha(I)$ of Y is a yes-instance

10. Prove the following: if X and Y are decision problems so that [5]
    - there is a polynomial time transformation from X to Y
    - $Y \in P$

    Then $X \in P$

Solution:
Suppose that $X \rightarrow_{poly} Y$, via the polynomial-time algorithm α, and in addition we know that $Y \in$ **P**, as witnessed by the polynomial-time algorithm β. Consider the algorithm γ

- Take as input some I of X
- Compute $\alpha(I)$ using algorithm α
- Decide whether $\alpha(I)$ is a yes-instance of Y using algorithm β, outputting 'yes' if it is, and 'no' it isn't

This algorithm solves X.
Let I be some instance of X of size n. The algorithms α and β take time at most $O(n^k)$ and $O(n^m)$ time units respectively. For an input n, the overall time takes On^{km}, so $X \in P$

11. Define what it means for a problem to be NP-complete. Prove that if Y is an NP-complete problem then **P=NP** if, and only if, $Y \in P$ [6]

Solution:

Suppose there is a problem $Y \in NP$ such that for any problem $X \in NP$, we have $X \rightarrow_{poly} Y$
Such problems are NP complete.
From the previous theorem

- If $Y \in NP$ then every problem $X \in NP$ is also in P; that is P=NP
- Conversely, suppose that P=NP, so, as $Y \in NP$, we must have that $Y \in P$

12. What is Cook's theorem? [2]

Solution:

P=NP iff $SAT \in P$

13. Carefully define the Satisfiability problem and explain how we measure the size of an instance [4]

Solution:

An instance of size n of the Satisfiability problem is a set of n clauses of literals involving the Boolean variables $X_1, X_2, ..., X_n$
A yes-instance of the satisfiability problem is an instance for which there exists a truth assignment t that makes at least 1 literal in every clause true