ELL781: Software Fundamentals for Computer Technology

# Minor Test II (SA)

## Maximum marks: 8

Section 1. Short Answer Questions

**NB: Please read all questions carefully. There may be subtle differences between what the question is asking for and the context in which things have been discussed during the lectures, and these have to be taken into account whilst answering.**

1. In class we proved the `NP`-completeness of `SAT` using the known `NP`-completeness of `CIRCUIT-SAT`. This question asks you to do the opposite. You need to give a proof of the `NP`-completeness of `CIRCUIT-SAT`, and you can assume that `SAT` is already known to be `NP`-complete. (This more closely reflects the historical chronology, whereby `SAT` was acutally the first problem to be established as `NP`-complete by Cook and Levin in 1971.)

   (a) Provide a complete proof by reduction that `CIRCUIT-SAT` is `NP`-complete. For simplicity, you may assume that any instance of `SAT` will make use of only 3 Boolean operators: $\wedge$, $\vee$, and $\neg$ (respectively `AND`, `OR`, and `NOT`). The other operators we saw in class ($\rightarrow$ and $\leftrightarrow$) can actually be written in terms of the first three.

(Hint: You will need to give an inductive proof that every instance of `SAT` can be converted to an equivalent instance of `CIRCUIT-SAT` in polynomial time. Please write out this proof carefully and clearly. Don't try to somehow invert the conversion process discussed in class; that won't work, and the process here will actually be much simpler than that!) **[6]**

(b) Use your conversion process described above to give a `CIRCUIT-SAT` instance which is equivalent to the following instance of `SAT`: $x_1 \wedge (\neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_4)$. **[2]**