CANDIDATE: please attach Student Support Unit sticker, if relevant

THE UNIVERSITY OF SUSSEX

MComp THIRD YEAR EXAMINATION
BSc FINAL YEAR EXAMINATION

LIMITS OF COMPUTATION
G5029

You can start this exam at a time of your choosing within a 24 hour window. Once started you will have a set exam duration in which to complete it (note: the assessment will close at end of the 24 hour window; start with sufficient time to complete).

If you have extra time due to Reasonable Adjustments this is additional to the exam duration below and has been added to your assessment on Canvas.

Date: 13 May 2021
24 Hour Window starts at: 09:30
Exam Duration: 3 hours (including time for scanning, collating, uploading)

Candidates should answer TWO questions out of THREE.
If all three questions are attempted only the first two answers will be marked.
Each question is worth 50 marks.

Write or type your answers on A4 paper, scan and save as a single PDF file and upload to Canvas.

Please make sure that your submission includes the following:

Your candidate number (Do not put your name on your paper)
The title of the module and the module code.

Read Academic Integrity Statement
You MAY access online materials, notes etc. during this examination. You must complete this assessment on your own and in your own words. DO NOT discuss this assessment with others before the end of its 24 hour window. By submitting this assessment you confirm that your assessment includes no instances of academic misconduct, for example plagiarism or collusion. Any instance of academic misconduct will be thoroughly investigated in accordance with our academic misconduct regulations.

1. This question is about various notions of effective computability.

   (a) Why do we not need to extend (change) the semantic function for the `WHILE`-language in order to interpret programs of the extended `WHILE`-language? [4 marks]

   (b) What kind of procedure calls does the extended `WHILE`-language provide? Explain carefully. [4 marks]

   (c) Explain the concept of indirect addressing used by the `RAM` computation model. Why is this needed at all? [5 marks]

   (d) What can you say about the relationship between `RAM`-decidable and `WHILE`-decidable problems about natural numbers? [4 marks]

   (e) Let the following `WHILE`-program `myprog` be as follows:

```
myprog read X {
       while X {
          Z := cons nil Z;
          X := tl X
       };
       Z := tl Z
 }
 write Z
```


   i. Assuming that we start counting variables from 0, give the program-as-data representation of `myprog`. [12 marks]

   ii. Is this `WHILE`-program written in *core* `WHILE` or *extended* `WHILE`? [2 marks]

   iii. What is $[\![\texttt{myprog}]\!]^{\texttt{WHILE}}(\ulcorner n \urcorner)$ for an arbitrary $n \in \mathbb{N}$? Recall that $\ulcorner n \urcorner$ the encoding of a natural number n as data. [3 marks]

   iv. What is $[\![\texttt{myprog}]\!]^{\texttt{WHILE}}(d)$ for an arbtirary $d \in \mathbb{D}$? [2 marks]

   (f) Assume c_1 is a compiler from `S` to `T` written in `L` . Assume further c_2 is a compiler from `L` to `M` written in `C`. What do we get if we compile c_1 using c_2? Be as precise as possible. [4 marks]

   (g) Can we add a new instruction to the instruction set of a standard Turing Machine, such that the resulting new Turing Machines, let's call them Turing Machines Plus, can decide more problems than the standard type of Turing Machines? Explain your answer. [5 marks]

   (h) Can we remove an instruction from the definition of standard Turing Machines, such that the resulting new Turing Machines, let's call them Turing Machines Minus, can decide fewer problems than the standard type of Turing Machines? Explain your answer. [5 marks]

2. This question is about decidability and semi-decidability, and self-referencing programs.

   (a) Let $A \subseteq \mathbb{D}$ be a problem (about binary trees). Assume there exists a `WHILE`-program `p` such that the following holds:

   $$[\![\texttt{p}]\!]^{\texttt{WHILE}}(\texttt{d}) = \texttt{true} \quad \text{if, and only if,} \quad \texttt{d} \in A.$$

   What can we conclude from the above for problem A? [4 marks]

   (b) Which of the following problems are `WHILE`-decidable? (No explanation needed.)

      i. Halting problem
      ii. Complement of the Halting problem
      iii. Travelling Salesman problem
      iv. Postman problem
      v. Tiling problem
      vi. The problem whether a natural number is a prime number
      vii. The problem whether a `WHILE`-program (as data) has the same behaviour as the `WHILE`-program `succ` (as given in lectures).

   [7 marks]

   (c) Which of the following statements about closure properties of problems (considered as subsets of $\mathbb{D}$) are always true (i.e. for any A, B, A_n, resp.)? (No explanation needed.)

      i. If A and B are both semi-decidable, their intersection, $A \cap B$, is also semi-decidable.
      ii. If A and B are both decidable, their union, $A \cup B$, is also decidable.
      iii. If A is semi-decidable, the complement of A is also semi-decidable.
      iv. If A is decidable, the problem $\{\, \langle a.a \rangle \in \mathbb{D} \mid a \in A \,\}$ is also decidable.
      v. If A is semi-decidable, the problem $\{\, \ulcorner l \urcorner \in \mathbb{D} \mid l \textit{ is a list that contains at least one element from } A \,\}$ is also semi-decidable. Recall that $\ulcorner l \urcorner$ denotes the encoding of a list as data.
      vi. If A_n is decidable for every $n \in \mathbb{N}$, the problem $\{\, d \in \mathbb{D} \mid d \in A_n \textit{ for some } n \in \mathbb{N} \,\}$ is also decidable.

   [6 marks]

   (d) Give a problem (set) that is semi-decidable but *not* decidable. [3 marks]

   (e) What proof technique (besides proof-by-contradiction) is used in the proof that the Halting Problem is undecidable? No explanation is required. [2 marks]

   (f) What proof technique (besides proof-by-contradiction) is used in the proof of Rice's Theorem? Explain what this technique is applied to. You don't have to explain the technique itself. [4 marks]

(g) Explain for the following sets $A \subseteq \texttt{WHILE}$-data whether they are WHILE-decidable. For each case, explain your answer. In cases where A is decidable this explanation should consist of a description of the decision procedure. Here $\ulcorner p \urcorner$ denotes the encoding of a WHILE-program as WHILE-data.

i. $A = \{ \ulcorner p \urcorner \mid$ WHILE-program p returns `nil` if its input encodes a WHILE-program that contains at least two assignment statements for the same variable $\}$ [4 marks]

ii. $A = \{ \ulcorner p \urcorner \mid \ulcorner p \urcorner$ contains at least two assignment statements for the same variable $\}$ [4 marks]

iii. $A = \{ \ulcorner p \urcorner \mid$ WHILE-program p returns `nil` for input `nil` $\}$ [4 marks]

(h) Consider a WHILE-program p with the following property:

$[\![p]\!]^{\texttt{WHILE}}(d) = \ulcorner p \urcorner$ for all $d \in \mathbb{D}$

Recall that $\ulcorner p \urcorner$ denotes the encoding of a WHILE-program p as data.

i. What sort of program is p referring to the specific property above? [2 marks]

ii. Explain carefully why programs with the above property are actually semantically well-defined, referencing results from the lectures. [6 marks]

iii. Explain briefly how this is related to reflection and meta-programming. [4 marks]

3. This question is about complexity.

(a) Let program `prog` be defined as follows:

```
prog read X {
   while X {
     X:= tl X;
     Y:= cons hd X Y
    }
 }
 write Y
```

i. Is the WHILE-program `prog` above in $\texttt{WHILE}^{time(8n+5)}$? Explain your answer carefully. [8 marks]

ii. Assume you have two WHILE-programs $\texttt{p} \in \texttt{WHILE}^{time(n^2+13)}$ and $\texttt{q} \in \texttt{WHILE}^{time(34n+12)}$ that have the same semantics. Under what conditions would it be better to use one or the other? [6 marks]

(b) The Travelling Salesman problem (TSP) is **NP**-complete.

i. Explain what is meant by this statement. As part of your answer explain also **NP**. [5 marks]

ii. Referring to TSP as specific example, explain to what extent approximation algorithms are a useful means to solve **NP**-complete problems (viewed as optimisation problems). [6 marks]

(c) Which of the following are true of the Cook-Levin Theorem? (No explanation required.)

i. It says something about the Satisfiability Problem.
ii. It is not a theorem but a thesis.
iii. It says that for polynomial decidability it does not matter which programming language we are using, as long as it is as powerful as Turing machines.
iv. It says the Travelling Salesman Problem is of exponential complexity.
v. It says that a certain problem is **NP**-complete.

[5 marks]

(d) Which of the following problems are *known* to be **NP**-complete?

i. 0-1 Knapsack problem
ii. Complement of the Halting problem
iii. Graph Colouring problem
iv. Tiling problem
v. Postman problem
vi. Factorisation Problem

[6 marks]

(e) In the following give a rough description of *how* you would prove the statement in question. You do not have to give a proof, but you are supposed to sketch the required plan, i.e. which activity you would need to carry out and/or which theorems you would use in which way. Be as precise as possible.

i. For some fixed problem A: how would you establish that A is in $\mathbf{LIN}^{\texttt{WHILE}}$? [4 marks]

ii. For some fixed problem A: how would you establish that A can't possibly be in $\mathbf{P}^{\texttt{WHILE}}$ unless $\mathbf{P}^{\texttt{WHILE}} = \mathbf{NP}^{\texttt{WHILE}}$? [5 marks]

(f) Consider the statement $\mathbf{BQP} = \mathbf{NP}$.

i. Explain what this statement means. [3 marks]

ii. What do we know about the validity of this statement? [2 marks]