G5029

# THE UNIVERSITY OF SUSSEX

## BSc FINAL YEAR and MComp THIRD YEAR EXAMINATION 2020 JUNE 2020 (A2)

## Limits of Computation

**Date: Wednesday, 3 June 2020**
**24 Hour Window starts at: 14.00**
**Exam Duration: 3 hours (including time for scanning and 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 your answers on A4 paper, scan and save as a single PDF file and upload to Canvas**
**PDF file name: candidate number_module**

**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

Academic Integrity Statement
You are reminded that you should not access online materials, notes etc. during this examination or discuss this assessment with others before the end of its 24 hour window.
By submitting this assessment you confirm that you have read the above Statement and are responsible for understanding and complying with Academic Misconduct regulations as they relate to this assessment.

1. This question is about WHILE and other notions of effective computability.

   (a) The WHILE-language has one built-in data type, the type of binary trees where leaves contain always atom $\mathrm{nil}$ (for pure WHILE). Explain briefly why this is sufficient to define the notion of computability. [4 marks]

   (b) Let foo be the following pure WHILE-program:

```
foo read L {
  out := cons nil nil;
  while L {
        X := hd L;
        if X
         { L:= tl L
         }
        else {
         out := nil;
         L:= nil
        }
    }
}
write out
```

   i. What is $[\![\texttt{foo}]\!]^{\texttt{WHILE}}(\ulcorner 1 \urcorner)$ and $[\![\texttt{foo}]\!]^{\texttt{WHILE}}(\ulcorner [1] \urcorner)$, respectively? No explanation required. [2 marks]

   ii. Give the semantics of the program foo, in other words give a precise definition of the partial function $[\![\texttt{foo}]\!]^{\texttt{WHILE}} : \mathbb{D} \to \mathbb{D}_\perp$. Do *not* re-explain or narrate the program. [6 marks]

   (c) Explain the *semantical* difference between a WHILE-expression and a WHILE-command. [4 marks]

   (d) Write a WHILE-program that has the following data representation:

```
[ 0,
  [
    [while,[var,0],[ [:=,1,[cons,[hd,[var,0]],[quote,nil]] ],
                     [:=,0,[tl,[var,0]] ]
                   ]
    ]
  ],
1]
```

   [10 marks]

   (e) Why is it important to have a computation model that supports programs-as-data? [6 marks]

   (f) Consider now computability of functions on natural numbers, i.e. functions of type $\mathbb{N} \to \mathbb{N}_\perp$.

i. For <u>each</u> of the following functions $f$ on natural numbers state whether they are `WHILE`-computable or not. Briefly explain your answer in a sentence.

A. $f(n) = \underbrace{n^{n^{\cdot^{\cdot^{n}}}}}_{n \text{ times}}$

B. $f(n) = \max K_n$ where
$K_n = \{k{\in}\mathbb{N} \mid [\![p]\!]^{\texttt{WHILE}}(\text{nil}){=}\ulcorner k \urcorner \text{ for program } p \text{ s.t. } |p| \leq n\}$

C. $f(n) = \begin{cases} 1 & \text{iff } [\![p]\!]^{\texttt{WHILE}}(p) \neq \bot \text{ where } p \text{ is the } n\text{-th } \texttt{WHILE}\text{-program} \\ 0 & \text{otherwise} \end{cases}$

D. $f(n) = \bot$ for all $n \in \mathbb{N}$

E. $f(n) = \min K_n$ where
$K_n = \{k{\in}\mathbb{N} \mid [\![p]\!]^{\texttt{WHILE}}(\text{nil}){=}\ulcorner k \urcorner \text{ for program } p \text{ s.t. } |p| \leq n\}$

[10 marks]

ii. Does computability of such functions depend on whether they are implemented in `WHILE` or `RAM`? Briefly explain your answer. [3 marks]

iii. If we removed the conditional statement (`if then else`) from the pure `WHILE`-language, could we still compute the same functions of type $\mathbb{N} \to \mathbb{N}_\bot$ that we did with the conditional? Explain your answer briefly. [5 marks]

2. This question is about semi-decidability, decidability, and various forms of reduction.

   (a) For <u>each</u> of the following problems state whether it is WHILE-semi-decidable or not. Explain your answer in one sentence.

      i. NAT: Given an element of $\mathbb{D}$, does it encode a natural number?

      ii. HALT (the halting problem for WHILE);

      iii. The complement of HALT;

      iv. The complement of the Traveling Salesman Problem.

      [8 marks]

   (b) For the following sets $A \subseteq$ WHILE-data state whether they are WHILE-decidable or undecidable and explain your answer. In cases where $A$ is decidable this explanation should consist of a description of the decision procedure. Recall that $\ulcorner p \urcorner$ denotes the encoding of WHILE-program $p$ in $\mathbb{D}$.

      i. $A = \{\, \ulcorner p \urcorner \mid p$ returns $nil$ if its input encodes a natural number $\}$ [4 marks]

      ii. $A = \{\, \ulcorner p \urcorner \mid \ulcorner p \urcorner = \ulcorner \texttt{read X } \{\, \texttt{while true } \{\}\} \texttt{ write X} \urcorner \,\}$. [4 marks]

      iii. $A = \{\, \ulcorner [p,k] \urcorner \mid p \in \mathcal{O}(n^k) \,\}$ [4 marks]

   (c) This question addresses the importance of *effective problem reduction* in computability theory.

      i. For <u>each</u> of the following statements say whether the given effective reductions from $A \subseteq \mathbb{D}$ to $B \subseteq \mathbb{D}$, $A \leq_{\text{rec}} B$, are true or false. Explain each answer briefly. In case where the answer is true this explanation should be a description of the reduction function.

         A. $\{\ulcorner 1 \urcorner, \ulcorner 2 \urcorner, \ulcorner 4 \urcorner\} \leq_{\text{rec}}$ HALT

         B. $\{\ulcorner n \urcorner \mid n \in \mathbb{N} \wedge n \text{ is odd}\} \leq_{\text{rec}} \{\ulcorner n \urcorner \mid n \in \mathbb{N} \wedge n \text{ is even}\}$

         C. $\{\text{nil}\} \leq_{\text{rec}}$ HALT

         D. HALT $\leq_{\text{rec}} \{\text{nil}\}$

         E. $\mathbb{D} \setminus$ HALT $\leq_{\text{rec}} \{\text{nil}\}$

         [10 marks]

      ii. Explain in one or two sentences only the difference between effective problem reduction and polynomial time reduction. [5 marks]

   (d) For <u>each</u> of the following statements about the recursion theorem say whether it is true or false. Explain your answer in one sentence.

      i. The recursion theorem implies that there exists a WHILE program that returns its own description as data whatever the input.

      ii. The recursion theorem does not apply to Java.

      iii. The recursion theorem implies that any programming language meeting the theorem's assumptions allows indirectly for recursive program definitions.

[6 marks]

(e) For each of the following problems state whether they are closed under the given operation. Explain your answer briefly.

i. If $A$ and $B$ are semi-decidable, is the set $C = A \cap B$, i.e. the intersection of $A$ and $B$, semi-decidable? [3 marks]

ii. if $A \subseteq \mathbb{N}$ is decidable, is the set of all identical pairs of numbers, i.e. $\{ (a, a) \mid a \in A \}$ decidable? [3 marks]

iii. if $A \subseteq \mathbb{N}$ is decidable, is the set of all numbers for which the busy-beaver function returns a number in $A$ that is smaller than $500$ decidable, i.e. $\{ a \mid a \in A \wedge BB(a) < 500 \}$ decidable? [3 marks]

3. This question is about complexity.

   (a) A function $f : \mathbb{N} \to \mathbb{N}$ is a *time bound* for an $\mathtt{L}$-program $p$ if, and only if, for all input $d \in \mathtt{L}{-}\mathrm{data}$ it holds that $time_p^{\mathtt{L}}(d) \leq f(|d|)$ where $|d|$ is the size of $d$.

      i. How does one measure the size of $d$, $|d|$, for $d \in \mathtt{WHILE}{-}\mathrm{data}$? [2 marks]

      ii. In which situation do we have $time_p^{\mathtt{L}}(d) = \bot$? [2 marks]

      iii. Explain why it makes a difference for the time bound of a WHILE-program $p$ on natural numbers whether $p$ uses natural numbers in unary or decimal representation, i.e. whether it encodes a number like $12$ as $\ulcorner 12 \urcorner$ or as a list $\ulcorner [1,2] \urcorner$ where $\ulcorner \_ \urcorner$ denotes the standard encoding of data as presented in lectures. [6 marks]

   (b) For <u>each</u> of the following statements, state whether they are (known to be) true, (known to be) false, or whether the answer is currently still unknown. Accordingly, write as answer either *true*, *false*, or *unknown*, respectively. Give a short reason for <u>each</u> answer (one sentence suffices).

      i. The *Halting Problem* is in **P**.

      ii. **NP** $\subseteq$ **EXP**

      iii. **P** $\subseteq$ **NP**

      iv. $\mathtt{WHILE} \preceq^{lintime} \mathtt{TM}$

      v. The *Satisfiability Problem* (SAT) is in **P**.

      vi. *Max-Cut* $\leq_P$ *Integer Programming*.

      vii. The *0-1 Knapsack Problem* is in **P** if one represents integers as unary numbers.

      viii. $\mathbf{NP}^{\mathtt{SRAM}} \neq \mathbf{NP}^{\mathtt{WHILE}}$

      ix. $\mathtt{WHILE} \preceq^{lintime} \mathtt{WH^1LE}$.

      x. (The optimisation problem version of) TSP is not in **APX**.

      [20 marks]

   (c) A sequence $s$ is a subsequence of a sequence $t$ if $s$ can be obtained from $t$ by deleting some (maybe none) elements in $t$. For instance, the sequence $1, 4, 7$, the sequence $2, 3, 4, 6$ and sequence $4, 7$ are all subsequences of $1, 2, 3, 4, 5, 6, 7$, respectively. Subsequence $1, 4, 7$ is obtained by deleting $2, 3, 5$ and $6$; subsequence $2, 3, 4, 6$ is obtained by deleting $1, 4, 5$ and $7$; subsequence $4, 7$ is obtained by deleting $1, 2, 3, 5$ and $6$.

   The *common subsequence problem* (CSP) is the following decision problem: given a set of finite sequences $s_1, \ldots, s_n$ (over a given finite alphabet), is there a sequence $s$ of length at least $K$ such that $s$ is a subsequence of *each* sequence $s_1, \ldots, s_n$?

   Show that CSP is in **NP**. [8 marks]

(d) Argue that $\mathbf{P}^{\texttt{WHILE}}$ is closed under complement, i.e. argue that the following holds:
If $A$ is in $\mathbf{P}^{\texttt{WHILE}}$, then $A$'s complement, $\mathbb{D} \setminus A$, is also in $\mathbf{P}^{\texttt{WHILE}}$. [6 marks]

(e) A sequence $s$ is a subsequence of a sequence $t$ if $s$ can be obtained from $t$ by deleting some (maybe none) elements in $t$ as set out in Question 3(c).
The *increasing subsequence problem* (ISP) is the following decision problem: given a sequence of integer numbers $s$ and a number $K$, is there a sorted subsequence of $s$ that has at least $K$ elements? By sorted we mean that the integers in the sequence appear in ascending order. For instance, for $s = 4, 1, 8, 2, 1, 1, 5$ and $K = 3$, the answer is `YES` as $1, 2, 5$ is an increasing subsequence of $s$; for $K = 4$ the answer is `YES`, as $1, 1, 1, 5$ is an increasing subsequence of $s$. For any $K \geq 5$ the answer would be `NO`.
Show that ISP $\leq_P$ CSP, i.e. that ISP is polynomially reducible to CSP – where CSP was explained in Question 3c) – by answering the following questions:

i. Describe the required reduction function $f$. [4 marks]

ii. Argue that the function $f$ you defined satisfies the condition of a reduction function, i.e. $x \in$ ISP iff $f(x) \in$ CSP (you will need to explain what $x$ is). [2 marks]

You don't have to show that the function is computable in polynomial time or total.