COMP3161/9164 22T3 Assignment 0
Proofs: Solutions

Marks : 15% of overall marks for the course.
A mark of x (out of 100) on this assignment will translate to .15x course marks.

Due date: **Wednesday, 6th of October 2021, 12 noon (Sydney Time)**

1 Task

In this assignment you will formally model a language of boolean computations using a variety of semantic techniques, including its syntax and sematics, and its compilation to various target languages.

Prepare your answers in one PDF file, preferably using LaTeX, where all prose is typeset. Figures may be drawn, but make sure they are legible. Please ensure all mathematics is formatted correctly. Some guidance will be posted on the course website.

Submit your PDF using the CSE give system, by typing the command

```
give cs3161 Proofs Proofs.pdf
```

or by using the CSE give web interface.

Part A (25 marks)

Consider the language of boolean expressions $\mathcal{P}$ containing just literals (`True`, `False`), parentheses, conjunction ($\wedge$) and negation ($\neg$):

$$\mathcal{P} = \{\texttt{True}, \texttt{False}, \neg\texttt{True}, \neg\texttt{False}, \texttt{True} \wedge \texttt{False}, \neg(\texttt{True} \wedge \texttt{False}), \dots\}$$

1. Write down a set of inference rules that define the set $\mathcal{P}$. The rules may be ambiguous. *(5 marks)*

$$\frac{}{\texttt{True}\ \mathcal{P}} \qquad \frac{}{\texttt{False}\ \mathcal{P}} \qquad \frac{e\ \mathcal{P}}{\neg e\ \mathcal{P}} \qquad \frac{e\ \mathcal{P}}{(e)\ \mathcal{P}} \qquad \frac{e\ \mathcal{P} \qquad e'\ \mathcal{P}}{e \wedge e'\ \mathcal{P}}$$

2. The operator $\neg$ has the highest precedence, and conjunction is right-associative. Define a set of simultaneous judgements to define the language without any ambiguity. *(5 marks)*

$$\frac{}{\texttt{True}\ \mathcal{P}_a} \qquad \frac{}{\texttt{False}\ \mathcal{P}_a} \qquad \frac{e\ \mathcal{P}_a}{\neg e\ \mathcal{P}_a} \qquad \frac{e\ \mathcal{P}_c}{(e)\ \mathcal{P}_a} \qquad \frac{e\ \mathcal{P}_a \qquad e'\ \mathcal{P}_c}{e \wedge e'\ \mathcal{P}_c}$$

$$\frac{e\ \mathcal{P}_a}{e\ \mathcal{P}_c}$$

3. Here is an *abstract syntax* $\mathcal{B}$ for the same language:

$$\mathcal{B} ::= \mathsf{Not}\ \mathcal{B} \mid \mathsf{And}\ \mathcal{B}\ \mathcal{B} \mid \mathsf{True} \mid \mathsf{False}$$

Write an inductive definition for the *parsing* relation connecting your unambiguous judgements to this abstract syntax. *(5 marks)*

$$\frac{}{\texttt{True}\ \mathcal{P}_a \leftrightarrow \mathsf{True}\ \mathcal{B}} \qquad \frac{}{\texttt{False}\ \mathcal{P}_a \leftrightarrow \mathsf{False}\ \mathcal{B}} \qquad \frac{e\ \mathcal{P}_a \leftrightarrow a\ \mathcal{B}}{\neg e\ \mathcal{P}_a \leftrightarrow (\mathsf{Not}\ a)\ \mathcal{B}}$$

$$\frac{e\ \mathcal{P}_d \leftrightarrow a\ \mathcal{B}}{(e)\ \mathcal{P}_a \leftrightarrow a\ \mathcal{B}} \qquad \frac{e\ \mathcal{P}_a \leftrightarrow a\ \mathcal{B} \qquad e'\ \mathcal{P}_d \leftrightarrow a'\ \mathcal{B}}{e \wedge e'\ \mathcal{P}_d \leftrightarrow (\mathsf{And}\ a\ a')\ \mathcal{B}} \qquad \frac{e\ \mathcal{P}_a \leftrightarrow a\ \mathcal{B}}{e\ \mathcal{P}_d \leftrightarrow a\ \mathcal{B}}$$

4. Here is a big-step semantics for the language $\mathcal{B}$

$$\frac{x \Downarrow \mathsf{True}}{\mathsf{Not}\ x \Downarrow \mathsf{False}} \qquad \frac{x \Downarrow \mathsf{False}}{\mathsf{Not}\ x \Downarrow \mathsf{True}} \qquad \frac{x \Downarrow \mathsf{False}}{\mathsf{And}\ x\ y \Downarrow \mathsf{False}} \qquad \frac{x \Downarrow \mathsf{True} \quad y \Downarrow v}{\mathsf{And}\ x\ y \Downarrow v}$$

$$\frac{}{\mathsf{True} \Downarrow \mathsf{True}} \qquad \frac{}{\mathsf{False} \Downarrow \mathsf{False}}$$

a) Show the evaluation of And (Not (And True False)) True with a derivation tree. *(5 marks)*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{True} \Downarrow \mathsf{True}} \qquad \dfrac{}{\mathsf{False} \Downarrow \mathsf{False}}}{\mathsf{And\ True\ False} \Downarrow \mathsf{False}}}{\mathsf{Not\ (And\ True\ False)} \Downarrow \mathsf{True}} \qquad \dfrac{}{\mathsf{True} \Downarrow \mathsf{True}}}{\mathsf{And\ (Not\ (And\ True\ False))\ True} \Downarrow \mathsf{True}}$$

b) Consider the following inference rule:

$$\frac{y \Downarrow \mathsf{False}}{\mathsf{And}\ x\ y \Downarrow \mathsf{False}}$$

If we assume that $x\ \mathcal{B}$ holds, is this rule derivable? Is it admissible? And if we *don't* assume that $x\ \mathcal{B}$ holds, how does this change your answers? Justify your answers. *(5 marks)*

> Regardless of whether we get to assume $x\ \mathcal{B}$, the rule is *not* derivable, because the inference rules for $\mathsf{And}\ x\ y$ both require a judgement of the form $x \Downarrow v$, and we have none at hand.
> If we assume $x\ \mathcal{B}$, the rule is admissible. It is provable by induction that for all x such that $x\ \mathcal{B}$, either $x \Downarrow \mathsf{True}$ or $x \Downarrow \mathsf{False}$. In both cases, it follows that $\mathsf{And}\ x\ y \Downarrow \mathsf{False}$ holds.
> If we do not assume $x\ \mathcal{B}$, the rule is *not* admissible. If x is a term outside the language $\mathcal{B}$, there will be no v such that $x \Downarrow v$. Therefore, none of the two inference rules for And are applicable.

Part B (20 marks)

Here is a small-step semantics for a language $\mathcal{L}$ with `True`, `False` and `if` expressions:

$$\frac{c \mapsto c'}{(\texttt{If}\ c\ t\ e) \mapsto (\texttt{If}\ c'\ t\ e)}(1) \quad \frac{}{(\texttt{If True}\ t\ e) \mapsto t}(2) \quad \frac{}{(\texttt{If False}\ t\ e) \mapsto e}(3)$$

1. Show the full evaluation of the term (`If True` (`If True False True`) `False`). *(5 marks)*

> Here's a pedantic derivation for ants:
>
> $$\dfrac{\dfrac{}{\texttt{If True (If True False True) False} \mapsto \texttt{If True False True}} \qquad \dfrac{\dfrac{}{\texttt{If True False True} \mapsto \texttt{False}} \qquad \dfrac{}{\texttt{False} \overset{\star}{\mapsto} \texttt{False}}}{\texttt{If True False True} \overset{\star}{\mapsto} \texttt{False}}}{\texttt{If True (If True False True) False} \overset{\star}{\mapsto} \texttt{False}}$$

2. Define an equivalent *big-step* semantics for $\mathcal{L}$. *(5 marks)*

> The evaluable expressions are all $\mathcal{L}$, and the values are $\{\texttt{True}, \texttt{False}\}$
>
> $$\frac{}{\texttt{True} \Downarrow \texttt{True}} \qquad \frac{}{\texttt{False} \Downarrow \texttt{False}} \qquad \frac{c \Downarrow \texttt{True} \qquad t \Downarrow v}{(\texttt{If}\ c\ t\ e) \Downarrow v}$$
>
> $$\frac{c \Downarrow \texttt{False} \qquad e \Downarrow v}{(\texttt{If}\ c\ t\ e) \Downarrow v}$$

3. Prove that if $e \Downarrow v$ then $e \overset{\star}{\mapsto} v$, where $\Downarrow$ is the big-step semantics you defined in the previous question, and $\overset{\star}{\mapsto}$ is the reflexive and transitive closure of $\mapsto$. Use rule induction on $e \Downarrow v$. *(10 marks)*

We will use the following facts about $\overset{\star}{\mapsto}$ without proof; they are both provable by fairly straightforward rule inductions.

Lemma 1 (Transitivity)**.** *If* $s \overset{\star}{\mapsto} t$ *and* $t \overset{\star}{\mapsto} u$ *then* $s \overset{\star}{\mapsto} u$.

Lemma 2 (Context closure)**.** *If* $c \overset{\star}{\mapsto} c'$ *then* $(\texttt{If}\ c\ t\ f) \overset{\star}{\mapsto} (\texttt{If}\ c'\ t\ f)$

...and now for the main result.

Proof. By rule induction on $e \Downarrow v$. One case for each rule, from top left to bottom right.

Case 1: We have to prove $\texttt{True} \overset{\star}{\mapsto} \texttt{True}$. This is immediate by reflexivity.

Case 2: We have to prove $\texttt{False} \overset{\star}{\mapsto} \texttt{False}$. This is immediate by reflexivity.

Case 3: We have as induction hypotheses (i) $c \overset{\star}{\mapsto} \texttt{True}$ and (ii) $t \overset{\star}{\mapsto} v$.

From (i) and context closure we have $(\texttt{If}\ c\ t\ f) \overset{\star}{\mapsto} (\texttt{If True}\ t\ f)$.

We can then construct the following derivation:

$$\dfrac{\dfrac{}{(\texttt{If True}\ t\ f) \mapsto t} \qquad \dfrac{}{t \overset{\star}{\mapsto} v}\ (\text{II})}{(\texttt{If True}\ t\ f) \overset{\star}{\mapsto} v}$$

By transitivity, we conclude that $(\texttt{If}\ c\ t\ f) \overset{\star}{\mapsto} v$

Case 4: We have as induction hypotheses (i) $c \overset{\star}{\mapsto} \texttt{False}$ and (ii) $f \overset{\star}{\mapsto} v$.

From (i) and context closure we have $(\texttt{If}\ c\ t\ f) \overset{\star}{\mapsto} (\texttt{If False}\ t\ f)$.

We can then construct the following derivation:

$$\dfrac{\dfrac{}{(\texttt{If True}\ t\ f) \mapsto f} \qquad \dfrac{}{f \overset{\star}{\mapsto} v}\ (\text{II})}{(\texttt{If False}\ t\ f) \Downarrow v}$$

By transitivity, we conclude that $(\texttt{If}\ c\ t\ f) \overset{\star}{\mapsto} v$

□

Part C (15 marks)

1. Define a recursive *compilation function* $c : \mathcal{B} \to \mathcal{L}$ which converts expressions in $\mathcal{B}$ to expressions in $\mathcal{L}$. *(5 marks)*

$$
\begin{array}{lcl}
c(\mathsf{True}) & = & \texttt{True} \\
c(\mathsf{False}) & = & \texttt{False} \\
c(\mathsf{Not}\ \mathsf{e}) & = & (\texttt{If}\ (c(\mathsf{e}))\ \texttt{False}\ \texttt{True}) \\
c(\mathsf{And}\ \mathsf{e}\ \mathsf{e}') & = & (\texttt{If}\ (c(\mathsf{e}))\ (c(\mathsf{e}'))\ \texttt{False})
\end{array}
$$

2. Prove that for all e, $\mathsf{e} \Downarrow \mathsf{v}$ implies $c(\mathsf{e}) \Downarrow \mathsf{v}$, by rule induction on the assumption that $\mathsf{e} \Downarrow \mathsf{v}$. *(10 marks)*

Proof. By rule induction on $\mathsf{e} \Downarrow v$. One case for each rule, from top left to bottom right.

Case 1: We have the induction hypothesis $c(\mathsf{e}) \Downarrow \texttt{True}$, and we must show that $c(\mathsf{Not}\ \mathsf{e}) = (\texttt{If}\ (c(\mathsf{e}))\ \texttt{False}\ \texttt{True}) \Downarrow \texttt{False}$. The derivation goes like this:

$$\dfrac{\dfrac{}{c(\mathsf{e}) \Downarrow \texttt{True}}\ (\text{IH}) \qquad \dfrac{}{\texttt{False} \Downarrow \texttt{False}}}{(\texttt{If}\ (c(\mathsf{e}))\ \texttt{False}\ \texttt{True}) \Downarrow \texttt{False}}$$

Case 2: Similar to case 1.

Case 3: We have the induction hypothesis $c(\mathsf{e}) \Downarrow \texttt{False}$, and we must show that $c(\mathsf{And}\ \mathsf{e}\ \mathsf{e}') = (\texttt{If}\ (c(\mathsf{e}))\ (c(\mathsf{e}'))\ \texttt{False} \Downarrow \texttt{True}$. The derivation goes like this:

$$\dfrac{\dfrac{}{c(\mathsf{e}) \Downarrow \texttt{False}}\ (\text{IH}) \qquad \dfrac{}{\texttt{False} \Downarrow \texttt{False}}}{(\texttt{If}\ (c(\mathsf{e}))\ (c(\mathsf{e}'))\ \texttt{False}) \Downarrow \texttt{True}}$$

Case 4: We have the induction hypotheses $c(\mathsf{e}) \Downarrow \texttt{True}$, and $c(\mathsf{e}') \Downarrow v$. We must show that $c(\mathsf{And}\ \mathsf{e}\ \mathsf{e}') = (\texttt{If}\ (c(\mathsf{e}))\ (c(\mathsf{e}'))\ \texttt{False}) \Downarrow v$. The derivation goes like this:

$$\dfrac{\dfrac{}{c(\mathsf{e}) \Downarrow \texttt{True}}\ (\text{IH}) \qquad \dfrac{}{c(\mathsf{e}') \Downarrow v}\ (\text{IH})}{(\texttt{If}\ (c(\mathsf{e}))\ (c(\mathsf{e}'))\ \texttt{False}) \Downarrow v}$$

Case 5: We have to show that $c(\mathsf{True}) = \texttt{True} \Downarrow \texttt{True}$. Here's how:

$$\dfrac{}{\texttt{True} \Downarrow \texttt{True}}$$

Case 6: As case 6, but for False.

□

Part D (40 marks)

1. Here is a term in λ-calculus:

$$(\lambda \mathrm{n}.\ \lambda \mathrm{f}.\ \lambda \mathrm{x}.\ (\mathrm{n}\ \mathrm{f}\ (\mathrm{f}\ \mathrm{x})))\ (\lambda \mathrm{f}.\ \lambda \mathrm{x}.\ \mathrm{f}\ \mathrm{x})$$

a) Fully β-reduce the above λ-term. Show all intermediate beta reduction steps. *(5 marks)*

$$
\begin{array}{ll}
 & (\lambda n.\ \lambda f.\ \lambda x.\ (n\ f\ (f\ x)))\ (\lambda f.\ \lambda x.\ f\ x) \\
\mapsto_\beta & (\lambda f.\ \lambda x.\ ((\lambda f.\ \lambda x.\ f\ x)\ f\ (f\ x))) \\
\mapsto_\beta & (\lambda f.\ \lambda x.\ ((\lambda x.\ f\ x)\ (f\ x))) \\
\mapsto_\beta & \lambda f.\ \lambda x.\ f\ (f\ x)
\end{array}
$$

b) Identify an η-reducible expression in the above (unreduced) term. *(5 marks)*

$\lambda x.\ f\ x$ will η-reduce to f.

2. Recall that in λ-calculus, booleans can be encoded as binary functions that return one of their arguments:

$$\mathbf{T} \equiv (\lambda x.\ \lambda y.\ x)$$

$$\mathbf{F} \equiv (\lambda x.\ \lambda y.\ y)$$

Either via $\mathcal{L}$ or directly, define a function $d : \mathcal{B} \to \lambda$ which converts expressions in $\mathcal{B}$ to λ-calculus. *(5 marks)*

We'll do it via $\mathcal{L}$. Hence we define $f : \mathcal{L} \to \lambda$ as follows:

$$
\begin{array}{lcl}
f(\texttt{True}) & = & \mathbf{T} \\
f(\texttt{False}) & = & \mathbf{F} \\
f(\texttt{If c t e}) & = & (f(c))\ (f(t))\ (f(e))
\end{array}
$$

We can then define $d = f \circ c$

3. Prove that for all e such that $e \Downarrow v$ it holds that $d(e) \equiv_{\alpha\beta\eta} v'$, where v' is the λ-calculus encoding of v. *(10 marks)*

Lemma 3. $\forall e\ v.\ e \Downarrow v \Rightarrow f(e) \equiv_{\alpha\beta\eta} f(v)$

Proof. By rule induction on $e \Downarrow v$. One case per rule, numbered from top left to bottom right.

Case 1: We need to show that $f(\texttt{True}) =\equiv_{\alpha\beta\eta} f(\texttt{True})$. This is immediate by reflexivity.

Case 2: We need to show that $f(\texttt{False}) \equiv_{\alpha\beta\eta} f(\texttt{False})$. This is immediate by reflexivity.

Case 3: Our inductive hypotheses state that $f(c) \equiv_{\alpha\beta\eta} f(\texttt{True})$ and $f(t) \equiv_{\alpha\beta\eta} f(v)$. We must show that $f(\texttt{If c t } e) \equiv_{\alpha\beta\eta} f(v)$. This is derived as follows:

$f(\texttt{If c t } e)$	$=$	(def)
$(f(c))\ (f(t))\ (f(e))$	$\equiv_{\alpha\beta\eta}$	(IH x2)
$(f(\texttt{True}))\ (f(v))\ (f(e))$	$=$	(def)
$\mathbf{T}\ (f(v))\ (f(e))$	$=$	(def)
$(\lambda x.\lambda y.x)\ (f(v))\ (f(e))$	$\equiv_{\alpha\beta\eta}$	(β-reduction)
$(\lambda y.f(v))\ (f(e))$	$\equiv_{\alpha\beta\eta}$	(β-reduction)
$f(v)$		

Case 4: Our inductive hypotheses state that $f(c) \equiv_{\alpha\beta\eta} f(\texttt{False})$ and $f(e) \equiv_{\alpha\beta\eta} f(v)$. We must show that $f(\texttt{If c t } e) \equiv_{\alpha\beta\eta} f(v)$. This is derived as follows:

$f(\texttt{If c t } e)$	$=$	(def)
$(f(c))\ (f(t))\ (f(e))$	$\equiv_{\alpha\beta\eta}$	(IH x2)
$(f(\texttt{False}))\ (f(t))\ (f(v))$	$=$	(def)
$\mathbf{F}\ (f(t))\ (f(v))$	$=$	(def)
$(\lambda x.\lambda y.y)\ (f(t))\ (f(v))$	$\equiv_{\alpha\beta\eta}$	(β-reduction)
$(\lambda y.y)\ (f(v))$	$\equiv_{\alpha\beta\eta}$	(β-reduction)
$f(v)$		

□

The desired theorem is a corollary of the above lemma, and the result from question C.2.

4. Suppose we added *unary local function definitions* to our language $\mathcal{P}$. Here's an

example in concrete syntax:

$$
\begin{array}{l}
\mathsf{let} \\
\quad g(x) = \neg x \\
\mathsf{in} \\
\quad g(\mathsf{True}) \\
\mathsf{end}
\end{array}
$$

We limit ourselves to non-recursive bindings (meaning functions can't call themselves), and first-order functions (meaning functions require boolean arguments).

a) Extend the abstract syntax for $\mathcal{B}$ from question A.3 so that it supports the features used in the above example. Use first-order abstract syntax with explicit strings. You don't have to extend the parsing relation. *(5 marks)*

> We assumes that $\mathcal{S}$ is the set of strings, and add the following rules to $\mathcal{B}$.
>
> $$\frac{\mathsf{x}\ \mathcal{S}}{(\mathsf{Var\ x})\ \mathcal{B}} \qquad \frac{\mathsf{f}\ \mathcal{S} \qquad \mathsf{e}\ \mathcal{B}}{(\mathsf{Call\ f\ e})\ \mathcal{B}} \qquad \frac{\mathsf{f}\ \mathcal{S} \qquad \mathsf{x}\ \mathcal{S} \qquad \mathsf{e}_1\ \mathcal{B} \qquad \mathsf{e}_2\ \mathcal{B}}{(\mathsf{Let\ f\ x\ e}_1\ \mathsf{e}_2)\ \mathcal{B}}$$

b) Define a scope-checking judgement, similar to the **ok** judgement from the lectures. It should check (a) that all names of variables and functions are used only within their scopes; and (b) that names used in variable (or function) position are indeed the names of variables (or functions). Hence, the following expressions should both be rejected:

$$
\begin{array}{l}
\mathsf{let} \\
\quad f(x) = \neg x \\
\mathsf{in} \\
\quad f(x) \\
\mathsf{end}
\end{array}
\qquad
\begin{array}{l}
\mathsf{let} \\
\quad f(x) = \neg x \\
\mathsf{in} \\
\quad f(f) \\
\mathsf{end}
\end{array}
\qquad
\begin{array}{l}
\mathsf{let} \\
\quad f(x) = x(\mathsf{True}) \\
\mathsf{in} \\
\quad f(\mathsf{False}) \\
\mathsf{end}
\end{array}
$$

The following are examples of things that should be accepted: nested definitions, and shadowed definitions.

```
let
  f(x) =
    let
      g(y) = ¬x ∧ y
    in
      g(x) ∧ ¬g(x)
    end
in
  f(False)
end
```

```
let
  f(x) = x
in
    let
      f(x) = f(x)
    in
      f(True)
    end
end
```

Note that the latter example is *not* a recursive call. *(10 marks)*

This solution treats function and variable names as belonging to separate namespaces (meaning a function name can't shadow a variable name, and vice versa).

$$\frac{}{\Gamma \vdash \mathsf{True}\ \mathbf{ok}} \qquad \frac{}{\Gamma \vdash \mathsf{False}\ \mathbf{ok}} \qquad \frac{\Gamma \vdash e\ \mathbf{ok}}{\Gamma \vdash (\mathsf{Not}\ e)\ \mathbf{ok}}$$

$$\frac{\Gamma \vdash e_1\ \mathbf{ok} \qquad \Gamma \vdash e_2\ \mathbf{ok}}{\Gamma \vdash (\mathsf{And}\ e_1\ e_2)\ \mathbf{ok}} \qquad \frac{\mathsf{x}\ \mathbf{var} \in \Gamma}{\Gamma \vdash (\mathsf{Var}\ \mathsf{x})\ \mathbf{ok}}$$

$$\frac{\mathsf{f}\ \mathbf{fun} \in \Gamma \qquad \Gamma \vdash e\ \mathbf{ok}}{\Gamma \vdash (\mathsf{Call}\ \mathsf{f}\ e)\ \mathbf{ok}} \qquad \frac{\Gamma, \mathsf{x}\ \mathbf{var} \vdash e_1\ \mathbf{ok} \qquad \Gamma, \mathsf{f}\ \mathbf{fun} \vdash e_2\ \mathbf{ok}}{\Gamma \vdash (\mathsf{Let}\ \mathsf{f}\ \mathsf{x}\ e_1\ e_2)\ \mathbf{ok}}$$

2 Late Penalty

You may submit up to five days (120 hours) late. Each day of lateness corresponds to a 5% reduction of your total mark. For example, if your assignment is worth 88% and you submit it two days late, you get 78%. If you submit it more than five days late, you get 0%.

Course staff cannot grant assignment extensions—if you need an extensions, you have to apply for special consideration through the standard procedure. More information here: `https://www.student.unsw.edu.au/special-consideration`

3 Plagiarism

Many students do not appear to understand what is regarded as plagiarism. This is no defense. Before submitting any work you should read and understand the UNSW plagiarism policy `https://student.unsw.edu.au/plagiarism`.

All work submitted for assessment must be entirely your own work. We regard unacknowledged copying of material, in whole or part, as an extremely serious offence. In this course submission of any work derived from another person, or solely or jointly written by and or with someone else, without clear and explicit acknowledgement, will be severely punished and may result in automatic failure for the course and a mark of zero for the course. Note this includes including unreferenced work from books, the internet, etc.

Do not provide or show your assessable work to any other person. Allowing another student to copy from you will, at the very least, result in zero for that assessment. If you knowingly provide or show your assessment work to another person for any reason, and work derived from it is subsequently submitted, you will be penalized, even if the work was submitted without your knowledge or consent. This will apply even if your work is submitted by a third party unknown to you. You should keep your work private until submissions have closed.

If you are unsure about whether certain activities would constitute plagiarism ask us before engaging in them!