Put your answers in a PDF file; name it if-then-else.pdf.

Warning: Other file formats will NOT receive any mark.

Question 1: 2 marks
         2: 2 marks
         3: 1 mark

Question 1

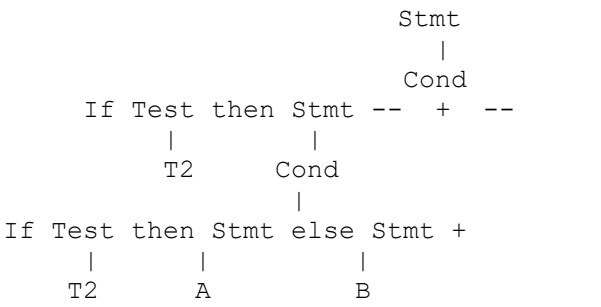
Most programming languages support if-then-else statements, with the "else" being optional. A long time ago, this was done carelessly and resulted in ambiguous grammars. Here is a simplified re-living of that time, with test conditions and statements replaced by terminal symbols to show the gist and avoid distractions.

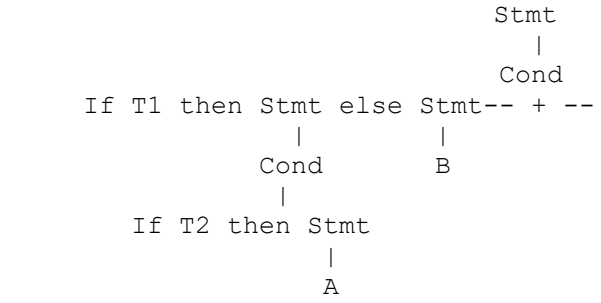
```
<stmt> ::= <cond> | "A" | "B" | "C"
<cond> ::= "if" <test> "then" <stmt>
                | "if" <test> "then" <stmt> "else" <stmt>
<test> ::= "T1" | "T2"
```

Give two different parse trees for

```
    if T1 then if T2 then A else B
```

Reminder: A parse tree, rather than an abstract syntax tree, is required.





Question 2

A way out is to add brackets, which is adopted by many imperative languages:

```
<stmt> ::= <cond> | "A" | "B" | "C"
<cond> ::= "if" <test> "then {" <stmt> "}"
                | "if" <test> "then {" <stmt> "} else {" <stmt> "}"
<test> ::= "T1" | "T2"
```

Add curry brackets to

```
    if T1 then if T2 then A else B
```

to fit this grammar. There are two versions, inspired by the two parse trees witnessed in Question 1; give both versions. (No need to draw the new parse trees.)

```
if T1 then {if T2 then {A} else {B}}

if T1 then {if T2 then {A}} else {B}
```

Question 3

Another way out is to make "else" compulsory, which is adopted by many functional languages:

```
<stmt> ::= <cond> | "A" | "B" | "C"
<cond> ::= "if" <test> "then" <stmt> "else" <stmt>
<test> ::= "T1" | "T2"
```

Give a parse tree for

```
    if T1 then if T2 then A else B else C
```

(and discover that you have only one choice).