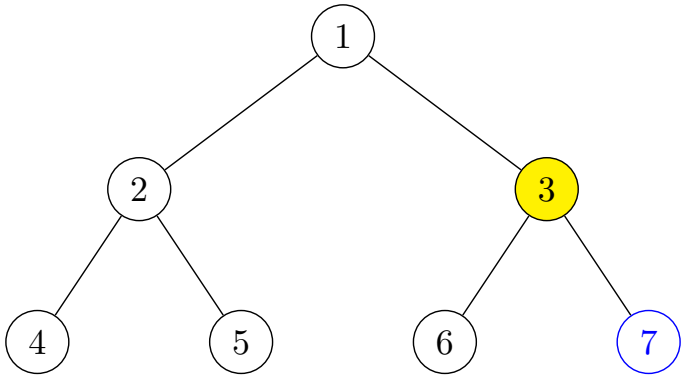
Growing Trees with Ti*k*Z

This is a **LaTeX task**. You must submit both your `.tex` source file and a compiled `.pdf` file. Your response will be assessed on both content and presentation.

Ti*k*Z is a language for drawing graphics that can be used within LaTeX.

In this task, we'll use Ti*k*Z to draw a tree. It should look something like this:



▶ Steps for you to do are marked with a triangle.

Setting up

▶ To start, create a new *standalone* document in a `.tex` file:

```
\documentclass{standalone}
\begin{document}

\end{document}
```

This document class provides only the bare necessities to make an image, and crops the resulting page.

▶ You may recall from *'Hello, World!'* in Module 1 that the section before `\begin{document}` is called the *preamble*, and is where we put configuration options. This time, we will add a command to import the Ti*k*Z package. Add this line to your preamble:

```
\usepackage{tikz}
```

You can think of this like a `#include` in C.

Drawing a picture

▶ To create our picture, we first need to open a `tikzpicture` environment. Use `\begin` and `\end` to create the environment, and put the following code inside:

```
\draw[fill=blue!40!white] (0,0) -- (4,0) -- (4,4) -- (0,4) -- (0,0);
```

Compile your document and see what you get!

▶ To learn more about the basics of Ti*k*Z, follow this tutorial.

Nodes and Edges

In Ti*k*Z, a node is simply some text at a point. For example,

```
\node (a) at (0, 0) {Hello! $2 + 2 = 4$};
```

The `(a)` is the name of the node, and will not be drawn. We can use this to refer to the node later:

```
\node at (0, 0) {Hello! $2 + 2 = 4$};
\node (b) [below=1cm of a] {Hi! $3 + 3 = 6$};
```

The node name can be left out if we don't need it. We'll look closer at nodes when we draw graphs later. Now, let's make a tree!

Trees

▶ Start a new standalone document with Ti*k*Z included, and place the following code in the body of the document:

```
\begin{tikzpicture}
\node {1}
    child {node {2}}
    child {node {3}};
\end{tikzpicture}
```

Here, we've used the `child` operation to give the node containing '1' a left and a right child. Currently, the nodes just show a number - let's give them an outline. As you've seen already, we can put options in square brackets `[]` to specify colours, placements and other properties of the elements in our picture. Multiple options can be separated by commas.

▶ Add the `draw` and `circle` properties to each node to give them a circular outline.

Manually adding options to each node gets tedious when we have a lot of nodes. Instead, we can specify to draw a circle around *every node* in the picture using a style:

```
\begin{tikzpicture}[every node/.style = {draw}]
```

▶ Add the style to draw a circular outline around every node.

Now, let's add some more nodes! We want to have 4 and 5 as the child nodes of 2, and 6 and 7 as the children of 3.

▶ Add the nodes 4, 5, 6 and 7.

Unfortunately, nodes 5 and 6 now overlap. To fix this, we will add more styles! As well as applying a style to every node, we can apply a style to a layer using `level x/.style`. Each layer of children

gets a level starting from 1, so the root's children are `level 1`, and its grandchildren are `level 2`. Is the root level 0? Try it and find out!

In this case, we want to use the `sibling distance` option. Unlike `draw`, this option takes a numeric value, which we assign with an `=` symbol, e.g. `sibling distance=1`. We can also specify a unit, e.g. `1cm`.

▶ Use level styles and `sibling distance` to fix the overlapping nodes.

Finally, we can add some decoration! Just like `sibling distance`, the `fill` option takes a parameter (in this case, a colour). You can also set the text colour by just specifying a colour as a standalone option (e.g. `blue`). Rather than applying this to all the nodes, we want to apply this to individual nodes, so we won't use a style.

▶ Use the fill and colour options to decorate the tree! Do some research and find some other options to style more nodes with.

Follow the steps given to produce a tree diagram using Ti*k*Z. Submit both the compiled PDF and your `.tex` code.

That is, produce a tree in Ti*k*Z similar to the one shown. In particular, you should have:

- Nodes numbered 1 to 7 with circular outlines,
- No overlapping nodes,
- Formatting applied with fill, colour and other style options,
- Use of styles for node outlines and level distances.

Your Ti*k*Z code should be well-formatted and readable, with good use of indentation and spacing.

You are free to create your own (more creative) tree.

Reminder. There is no `solution` environment in this template. This is intentional; you should *not* append the desired graph to this template file.

Instead, your PDF submission should be a `standalone` document, containing only the diagram.