CIS*2750
Assignment 2

Module 2 - modifying SVG structs

```
void setAttribute(SVG* img, elementType elemType, int elemIndex, Attribute* newAttribute);
```

This is a generic function that sets the attribute of an `SVG`, `Circle`, `Rectangle`, `Path`, or `Group`. For example, you might decide to add a `fill` attribute to a `Rectangle`, change the radius of a `Circle`, etc..

Note that while this function operates only on elements that are immediate children on the `svg` element in the original SVG file. In other words, it can set an attribute of a `Rectangle` in `SVG->rectangles`, or a `Group` in `SVG->groups`, but not attributes of a Rectangle belonging to a `Group` in `SVG->groups`.

For example, if we create an `SVG` from `quad01.svg`, we can change the attributes of any of the three groups, or the attributes of the path at the bottom of the file. However, we cannot use this function to access elements deeper in the XML tree, e.g. the rectangle or path within the first group.

The arguments are:
- `ima`: Pointer to the SVG struct that you want to modify
- `elemType`: Value indicating which struct we are modifying, e.g. `SVG`, `Circle`, etc. See the `elementType` definition in A2 header. We will use it so that we know which of the `SVG` lists we need to index into.
- `elemIndex`: index of the element that we want to modify in the relevant list within an `SVG` struct. This argument is ignored if we are modifying an `SVG` struct itself.
- `newAttribute`: the new attribute. The name and value of the attribute must be valid strings in the appropriate format - e.g. name is `cx` if you want to set the `Circle` centre, `d` is you want to set the `Path` data, `fill` if you want to set the `Rectangle` fill, etc..

The function should behave as follows:
- Use `elemType` and `elemIndex` to get the appropriate element.
- If the name of the attribute corresponds to a field of the struct (`cx` or `r` for `Circle`, `d` for Path, `x` or `y` for `Rectangle`, etc.), set the appropriate field of the corresponding struct to the value of the new attribute, and free the `newAttribute` struct. Apply type conversion as necessary. Keep in mind that this function does not change the units of the element it is modifying, so you do not need to update that field.
- Otherwise:
  - If the attribute with the specified name exists in the `otherAttributes` list of the relevant element, update the value on the Attribute in the list to the new value, and free the `newAttribute` struct.
  - If the attribute with the specified name does not exist in the `otherAttributes` list of the relevant element, append the new attribute to that list.
- Return values:
  - If the function successfully updated / added an attribute, return `true` to indicate success.
  - If the function failed to to the update / insert for nay reason - null elements, index is out of bounds, etc.. the function should return `false`.
    NOTE: if the function returns false, is **does not** need to free `newAttribute`. We will let the caller decide how to handle this error. This will include letting the error decide what to do with `newAttribute`.

For the following examples, let's assume that we crated an `SVG` from the file `rects.svg`:

- If we want to update the width of the `svg` element (i.e. `SVG` struct itself) to 6cm, we call `setAttribute(image, SVG_IMG, 0, newAttr)` with `newAttr->name = width` and `newAttr->value`

`= 6cm.` Attribute with name `width` exists in the `otherAttributes` list of that image, so we change its value to `6cm`. Since we are modifying an `SVG` struct, the index `0` is ignored.

- If we want to add black fill to the 1st rectangle (the one following `desc` in the original file), we call `setAttribute(image, RECT, 0, newAttr)` with `newAttr->name = fill` and `newAttr->value = black.` We search the `otherAttributes` list of the rectangle with index 0. Attribute with name `fill` does not exists in the `otherAttributes` list of that rectangle, so we add a new attribute to the `otherAttributes` list of that rectangle.
- If we want to change the width of the 2nd rectangle to 2cm, we call `setAttribute(image, RECT, 1, newAttr)` with `newAttr->name = width` and `newAttr->value = 2.` Since width has a dedicated a field in the Rectangle, we do not need to search the list - we simply change `Rectangle->width` of that rectangle to 2.
- If we want to change the fill of the 5th rectangle to red, we call `setAttribute(image, RECT, 4, newAttr)` with `newAttr->name = fill` and `newAttr->value = red.` We search the `otherAttributes` list of the rectangle with index 4. Attribute with name `fill` exists in the `otherAttributes` list of that rectangle, so we change its value to `red`.

If any of the arguments are invalid - NULL pointers, invalid element type, etc. - the function must do nothing. Make sure you don't create memory leaks when doing error handling. Also, make sure you don't leak memory when updating existing attributes.

```
void addComponent(SVG* img, elementType type, void* newElement);
```

This is a generic function for adding a new component to an existing `SVG` struct. New components are always added at the end of the component list. This function only needs to handle `Circles`, `Rectangles`, and `Paths`.

The arguments are:
- `img`: Pointer to the `SVG` we are modifying
- `type`: Value indicating which struct we are modifying, i.e. `CIRC`, `RECT`, or `PATH`. We will use it so that we know how to dereference the generic `newComponent` pointer.
- `newComponent`: the new component.

This function will append the new component to the end of the appropriate list in and `SVG` struct, after checking the `elemType` variable. It must do nothing if any of the arguments are invalid.