Exercise 3
Callan Moore

Exercise 3.1

Lua does not have switch cases and so the closest it can come to mimic this is to use multiple condition statements using ‘elseif’.

Exercise 3.2

```
-- Unconditional While Loop
function whileloop()
   while (true) do
      print("loop")
   end
End

-- unconditional repeat-until loop
function repeatloop()
   repeat
      print("loop")
   until false
end

-- unconditional
function gotoloop()
   ::loop::
   print("loop")
   goto loop
end

-- Unconditional For Loop
function forloop()
   for i = 1, math.huge do
      print("loop")
   end
end
```

I prefer the first While loop as it is most similar to C++ syntax which is my preferred language as I have the most experience with it.

Exercise 3.3

There are not many cases where the condition check needs to compute after the body of the function. In the small chance that this does happen an easy fix is to just duplicate the body of the loop in front of it so that it runs it once and then can check the condition for an ordinary while loop.

Exercise 3.4

```
function concatenate(...)
        local str = "";
```

```
        for i,v in ipairs{...} do
                s = s .. v
        end
         return s
end
```

Exercise 3.5

```
function ArrayPrint(...)
        for j, k in pairs(…) do
                print(j, k)
        end
end
```

Pros:

- The function is prebuilt and for standard use saves time of implementation of the function yourself.

Cons:

- Has ‘i = 1’ as a built in default so you can’t change the starting value.
- Will not return the index value

Exercise 3.6

PrintAllButFirst

```
function PrintAllButFirst(...)
        local tab = {...}
         for i = 2, #tab do
                print(tab[i])
        end
end
```