Instructions:

- Write the quiz solution on blank paper, numbering each of the questions and ideally answering the questions in order. Make sure your name and student number are clearly written at the top of the paper.
- When finished with the quiz. Take photos/scans (using your phone is fine) and format the photos in a single document (e.g. paste in a Word document, or anything else). Then create a PDF file not bigger than 10MB and upload to BB using the quiz submission link.
- As with all MATH2504 assessment, create a short voice recording indicating the work is your own and stating that you followed all of the guidelines (if this is the case). Upload the voice recording as an additional file.
- The duration of the quiz is 50 minutes + 10 minutes reading time + 20 more minutes allocating to formatting the solution and creating the voice recording. Thus 80 minutes in total.
- Join the course Zoom link at the start of the quiz (actual quiz is Aug 31, 6:00pm BNE time). A link to the actual quiz will be provided. You can then ask questions (only during) the first 10 minutes (reading time) via private chat on Zoom. It is recommended you stay on Zoom for the duration of the quiz (until final upload at 7:20pm). This is in case there are announcements or comments. There is no need for you to turn on your camera or mic.
- You are not allowed to communicate with **anyone** during the quiz. The only communication allowed is asking questions to the instructor via Zoom private chat during the first 10 minutes.
- You are not allowed to run Julia or any other computational software during the quiz. You may use a hand calculator (or calculator on your phone), but not more.
- You are not allowed to use any material except for the course material directly from the course website (Units 1–3, Practicals A–D, HW1-HW2). That is, feel free to use your web-browser to look at the course materials during the practice quiz, however you are not allowed to use **any** other written material or any other material from the web.
- In your voice recording, if it is the case, you should clearly state that you didn't communicate with anyone, that you didn't run Julia or any other software, and that you didn't use any other material except for (perhaps) material from the course website.
- In case of exceptional circumstances (mishap with upload etc...), write the course coordinator. Otherwise, late submissions will not be accepted.

Each item is worth 15 points.
There are 105 total points total but the maximal grade is 100.

Quiz questions on next page...

Question 1: Consider the Julia function below where `???` is a missing piece of code:

```
 1: function ok_brackets(str::String)
 2:     level = 0
 3:     for c in str
 4:         if c == ’(’
 5:             level += 1
 6:                                         #This line is intentionally empty
 7:         elseif c == ’)’
 8:             level -= 1
 9:         end
10:         level < 0 && return false
11:     end
12:     return ???
13: end
```

The function gets as input string expressions such as `"2+(2*x +(5*x^2+7))+ 4 - (3+x)^2"`. It returns `true` if the brackets are balanced and returns `false` if the brackets are not balanced.

For example for input such as the string above the result needs to be `true` since every opening bracket has a corresponding closing bracket and every closing bracket has a corresponding opening bracket. However, for input such as `"(5x+3("` the result should be `false`.

1a: Determine an expression in place of `???` of line 12 so that the function operates correctly.

1b: Assume you call this function with an input string of length n where n is an even number. What is the maximal number of times that line 8 can be executed? Explain the rational for your answer.

1c: You now wish to make a related function called `max_level` which operates on an input string similarly to `ok_brackets`. However this new function has a different return value. It returns the maximal depth of the expression and returns -1 if the orientation of the brackets is not ok. For example, your new function will return `-1` for an input of `"(5x+3("` and will return `3` for an input of `"2+(2*x +(5*x^2+(7)))+ 4 - (3+x)^2"`. Determine code that would create such a function by modifying **only** lines from 1 (for the name of the function), 2, 6, 10, and 12. Specify the code for these modified lines.

Question 2: Assume that you want to store a “secret” 8 letter word via double precision (`Float64`) variables. You do this by treating the double precision variable as a collection of 8 bytes where each byte is a character. Given a floating point number, `fn`, you use this function to print out the “secret“ word:

```
1: function decipher_float(fn::Float64)
2:     x = reinterpret(Int64,fn)
3:     mask = UInt64(0xFF)
4:     for i in 1:8:64
5:         c = Char((x & mask) >> (i-1))
6:         print(c)
7:         mask = mask << 8
8:     end
9: end
```

The use of the in-built `reinterpret()` function in line 2, simply creates an `Int64` variable `x` that uses the same bits as in the `Float64` variable `fn`. The rest of the code works by pulling out the relevant bits out of `x`, interpreting them as characters, and printing.

2a: Say the range in line 4 was changed from `1:8:64` to `1:8`. What would you change in other lines of the function to keep the same output? As a guide remember that the elements represented via `1:8:64` are $1, 9, 17, 25, \ldots, 57$ and those represented via `1:8` are $1, 2, 3, \ldots, 8$.

2b: Returning now to the original range in line 4 (`1:8:64`) suggest a replacement for lines 5–7 that only uses ‘`>>`’ once and does not use ‘`<<`’ at all.

Question 3: The following function is supposed to find the maximum of an $n \times n$ square matrix where n is a power of 2. Lines 3 and 4 simply check that the input is valid and can be ignored.

```
 1: function find_max(A::Matrix{Float64})
 2:     n, m = size(A)
 3:     @assert n == m    #make sure matrix is square
 4:     @assert ispow2(n) #make sure dimension is a power of 2
 5:     n == 1 && return 0
 6:     hn = n >> 1
 7:     lh, uh = 1:hn, (hn+1):n
 8:     return max( find_max(A[lh,lh]),
 9:                 find_max(A[lh,uh]),
10:                 find_max(A[uh,lh]),
11:                 find_max(A[uh,uh])  )
12: end
```

3a: There is a mistake in one of the lines of the function (only a single line). What is the mistake and how should it be fixed?

3b: Assume you call the function on a matrix of size n (which is a power of 2). In total (including the recursive calls), how many times will line 5 be executed?