Styles of exam questions to revise

The aim here is to help you guide your revision - when you're revising try to think of ways we might ask questions about the material. Looking at past exams is helpful, but do remember that this year is a bit different, and some styles of questions from previous years aren't usable in this year's open-book online exams.

Some categories of styles of questions:

- tracing code **(this one is not very usable this year!)**
- reproducing code we've seen in lecture notes (e.g. binary search, bubble sort, etc) **(this one is not very usable this year!)**
- describing in words what code does
- explaining why a piece of code does what it does
- with provided code and an output, give me an example input that would have generated that output
- write code (usually a function) that does some specified task
- find the bugs in a piece of code, and explain why they are a problem and how to fix them
- writing code that includes defensive error-checking or exception catching or both
- giving and justifying the big-O complexity of a piece of code
- giving and justifyng the big-O complexities of combined pieces of code
- a larger coding question that involves designing a data structure, reading into that structure from a file or string, and writing functions that operate on that structure

While you **cannot collaborate during the exam**, I encourage you to share practice problems that you devise with each other ahead of time during your revision.

Let's look at a few examples of these types of questions:

Describing in words what code does

```
def function_a(dict_1):
    x = 0
    for key in dict_1:
        x = x+dict_1[key]
    return x
```

Describe in words what `function_a` does.

Sample answer: `function_a` takes a dictionary as a parameter. It adds up all the values in the dictionary and returns that sum.

```
def function_b(dict_1, f):
    x = 0
    my_list = []
    for key in dict_1:
        if type(key) == type (''):
           my_list.append(f(dict_1[key]))
    return my_list
```

Describe in words what `function_b` does.

Sample answer: `function_b` is a higher order function that takes a dictionary and a function as arguments. It traverses the dictionary, and for each key in the dictionary that is a string, it applies the argument function to the value associated with that key and adds it to a list. Finally, it returns that list.

Explaining why a piece of code does what it does

Similar to above, but might include particular questions like: why does `function_b` not change the dictionary that is passed to it as an argument? What happens if we pass `function_a` a list instead of a dictionar as an argument? Why can we not access `my_list` directly outside of `function_b`?

With provided code and an output, give me an example input that would have generated that output

I have defined the function `my_func` as

```
def my_func(num):
    return num + 4
```

and have run `function_b(my_dict, my_func)` which returned the list `[4, 0, 7]` give an example `my_dict` argument that would have produced this output.

Write code (usually a function) that does some specified task

Define a function that takes a list as an argument, sums the integer values in that list, and returns that sum. You do not need to include error-checking.

Define a function that takes a dictionary and a list as arguments and returns a new dictionary that is a subset of the argument dictionary, and includes as keys only values that are also found in the argument list. Keys in the new dictionary should be mapped to that same values they are mapped to in the argument dictionary. Include error-checking so that if there is an element in the list that is not a key in the argument dictionary a warning is printed but the function still returns a dictionary as specified above.

Find the bugs in a piece of code, and explain why they are a problem and how to fix them

We've seem a few eamples of these in quizzes and labs - particularly for binary search in the searching lab.

Writing code that includes defensive error-checking or exception catching or both

As with the previous code-writing examples, but with a specific mention of defensive error checking or exception catching.

Giving and justifying the big-O complexity of a piece of code

What is the big-O complexity of `function_a`? Briefly justify your answer.

What is the big-O complexity of a simple search? Briefly justify your answer.

What is the big-O complexity of bubble sort? Briefly justify your answer.

Giving and justifyng the big-O complexities of combined pieces of code

Say I run `function_b` and then run a simple search on the output. What is the resulting big-O complexity? Briefly justify your answer. You may assume the big-O complexity of the function that is an argument to `function_b` is O(1).

Say I run `function_b` and then run bubble sort on the output. What is the resulting big-O complexity? Briefly justify your answer. You may assume the big-O complexity of the function that is an argument to `function_b` is O(1).

A larger coding question that involves designing a data structure, reading into that structure from a file or string, and writing functions that operate on that structure

Here we can find examples in past exams - the cards exercise in lab cycle 8 is based on part of one.