Part B – Questions

Question 1

The main reason that a worker is allocated different jobs each time the application is run, is because of time slicing. Each thread is given a short amount of time to run on the processor. However, the point at which a thread is given control depend on the priority that the application has as there are other, more important processes taking place within the operating system that need to execute at the same time. Due to us not knowing what processes need to happen when, each time the application is run, each thread may be allocated a different time slice, thus meaning, the job allocated each time will be different.

Question 2

The purpose of the `resourceStackChangeLock` object is to make sure that all threads are prevented from accessing the resource stack, whilst changes such as popping resources off the stack, pushing resources onto the stack and retrieving the size of the stack, are taking place. The `sufficientResourceCondition` object is there to notify and control the access of the threads to the stack. If there are not the required amount of resources for a job on the stack, this object tells that thread to wait. When resources are pushed onto the stack, the object signals to all threads telling them to check resource availability once more.

Question 3

A deadlock would occur because at first, a thread would see the current size of the stack and if there are sufficient resources available, it will attempt to pop resources off. If you have two or more threads accessing the stack at the same time, there may not be sufficient resources to allocate to all of them at the same time. Therefore, you would end up with 2 or more threads not having the required number of resources, thus would be waiting for each other to return resources to the stack which of course, is impossible until the job has been completed which with insufficient resources, cannot happen.