Assignment 18

R1: 9
R2: 9

	HAS R1	MAX R1	HAS R2	MAX R2
A	0	**6**	0	**8**
B	0	**8**	0	**2**
C	0	**4**	0	**1**

1.

```
A: alloc_R1(2)
A: alloc_R2(1)
B: alloc_R1(3)
B: alloc_R2(1)
C: alloc_R1(2)
C: alloc_R2(1)
```

	HAS R1	MAX R1	HAS R2	MAX R2
A	2	**6**	1	**8**
B	3	**8**	1	**2**
C	2	**4**	1	**1**

```
C: alloc_R1(2): C finishes
```

	HAS R1	MAX R1	HAS R2	MAX R2
A	2	**6**	1	**8**
B	3	**8**	1	**2**
C	-	**-**	-	**-**

```
A: alloc_R1(4), A: alloc_R2(7): A finishes
```

	HAS R1	MAX R1	HAS R2	MAX R2
A	-	**-**	-	**-**
B	3	**8**	1	**2**
C	-	**-**	-	**-**

```
B: alloc_R1(5), alloc_R2(1): B finishes
```

	HAS R1	MAX R1	HAS R2	MAX R2
A	-	**-**	-	**-**
B	-	**-**	-	**-**
C	-	**-**	-	**-**

Therefore, the state described after the initial allocation is a safe state.

2.

A: release_R1(1)
B: alloc_R1(2)

Run Banker's:

```
E = 9 9
A = 1 6
C = 1 1
    5 1
    2 1
N = 5 7
    3 1
    2 1
```

	HAS R1	MAX R1	HAS R2	MAX R2
A	1	**6**	1	**8**
B	5	**8**	1	**2**
C	2	**4**	1	**1**

We see that any further allocation from R1 will not fulfill the need of any of the three processes, while the needs of B and C from resource R2 could be satisfied, but that's irrelevant as none of the processes can finish. Therefore this state is unsafe and B request should be denied and that process put into waiting.

3.

	HAS R1	MAX R1	HAS R2	MAX R2
A	1	**6**	1	**8**
B	3	**8**	1	**2**
C	-	**-**	-	**-**

Run Banker's:

```
E = 9 9
A = 5 6
C = 1 1
    3 1
N = 5 7
    5 1
```

As we can see, B requires 5 of R1 and 1 of R2 to complete. Therefore granting B it's requested amount of 2 would result in a safe state from which B could finish. Conversely A could also finish before B without any release of resources, but B is further along so logically that one should be allowed to finish.