Task 1.1

i	j	Change(i)	Vars(j)	i intf w j
s1	s2	$r1, i1$	$n, r2, i2$	no
s2	s1	$r2, i2$	$n, r1, i1$	no

Task 1.2

```
                                                                  $\{n \geq 0\}$
$i1 := 0;$
$r1 := 0;$                                       $\{n \geq 0 \wedge i1 = 0 \wedge r1 = 0\}$
$\{\textbf{inv } r1 = sum(0, i1) \wedge i1 \leq n/2\}$
while $(i1 < n/2)\{$
  $r1 := r1 + 1;$
  $i1 := i1 + 1;$
$\}$                                                      $\{r1 = sum(0, n/2)\}$
$i2 := n/2;$
$r2 := 0;$                              $\{r1 = sum(0, n/2) \wedge i2 = n/2 \wedge r2 = 0\}$
$\{\textbf{inv } r1 = sum(0, n/2) \wedge r2 = sum(n/2, i2) \wedge i2 \leq n\}$
while $(i2 < n)\{$
  $r2 := r2 + i2;$
  $i2 := i2 + 1;$
$\}$                                    $\{r1 = sum(0, n/2) \wedge r2 = sum(n/2, n)\}$
                                                  $\Rightarrow \{r1 + r2 = sum(0, n)\}$
$r := r1 + r2;$
                                                              $\{r = sum(0, n)\}$
```

Task 1.3

```
                                                    {$n \geq 0$}
[                                                   {$p_1 \equiv n \geq 0$}
$i1 := 0;$
$r1 := 0;$
{**inv** $r1 = sum(0, i1) \wedge i1 \leq n/2$}
while ($i1 < n/2$){
  $r1 := r1 + i1;$
  $i1 := i1 + 1;$
}                                                   {$q_1 \equiv r1 = sum(0, n/2)$}
||                                                  {$p_2 \equiv n \geq 0$}
$i2 := n/2;$
$r2 := 0;$
{**inv** $r2 = sum(n/2, i2) \wedge i2 \leq n$}
while ($i2 < n$){
  $r2 := r2 + i2;$
  $i2 := i2 + 1;$
}                                                   {$q_2 \equiv r2 = sum(n/2, n)$}
]                                 {$q_1 \wedge q_2$} $\Rightarrow$ {$r1 + r2 = sum(0, n)$}
$r := r1 + r2$
                                                    {$r = sum(0, n)$}
```

i	j	Change(i)	Vars(j)	FV(j)	i intf w j	i intf w cond j
s1	s2	$r1, i1$	$n, r2, i2$	$r2, n$	no	no
s2	s1	$r2, i2$	$n, r1, i1$	$r1, n$	no	no

Task 2.1

```
                                                            $\{n \geq 0 \land i1 = 0 \land r1 = 0\}$
[                                                           $\{n/2 \leq i2 \leq n \land i1 = 0 \land r2 = sum(n/2, i2) \land r1 = 0\}$
{inv $i1 \leq n/2 \land n/2 \leq i2 \leq n \land r2 = sum(n/2, i2)$}
while $(i1 < n/2)\{$
   $< r1 := r1 + i1;$
  $i1 := i1 + 1; >$
}                                                           $\{r1 = sum(0, n/2)\}$
||                                                          $\{0 \leq i1 \leq n/2 \land i2 = n/2 \land r1 = sum(0, i1) \land r2 = 0\}$
{inv $i2 \leq n \land i1 \leq n/2 \land r1 = sum(0, i1)$}
while $(i2 < n)\{$
  $r2 := r2 + i2;$
  $i2 := i2 + 1;$
}                                                           $\{r2 = sum(n/2, n)\}$
{inv $i1 \leq n/2 \land r1 = sum(0, i1) \land r2 = sum(n/2, n)$}
while $i1 < n/2\{skip\}$                                    $\{r1 = sum(0, n/2) \land r2 = sum(n/2, n)\}$
                                                            $\Rightarrow \{r1 + r2 = sum(0, n)\}$
$r := r1 + r2$
]
                                                            $\{r = sum(0, n)\}$
```

Task 2.2

These two proof outlines are interference-free because one thread doesn't invalidate any condition needed by the other thread. Every atomic statement $\{r\} < s > \{...\}$ in s_1^* doesn't interfere with the proof outline $\{p2\}\ s_2^*\ \{q2\}$. So, the proof outline $\{p1\}\ s_1^*\ \{q1\}$ does not interfere with the other proof outline $\{p2\}\ s_2^*\ \{q2\}$.

Task 3.1

Instead of {while $(i1 < n/2)\{skip\}$; $r := r1 + r2$}, I just use {await $i1 = n/2$ then $r := r1 + r2$}

Task 3.2

It's preferable to use await instead of a loop because the loop is wasteful. While it does work, it repeatedly checks to see if the condition is true. With await, once it checks the condition and it's false, it gets blocked, and waits until the condition is true. So, other threads are run nondeterministically and only goes back to the thread with the await if the condition becomes true.

Task 4.1

(a) The problem is that, the "wait" in thread 2, is no longer doing what it's intended. It no longer waits for the other thread, thread 1, to run and finish because it already satisfies the condition even before running thread 1. If thread 2 starts from the beginning, it can finish the whole thing because of the invalid initial conditions in the precondition as well as the initial state σ. So, with the given σ, we get $r = r1 + r2 = 523752 + sum(18, 26)$, just from running and finishing thread 2. It then goes to thread 1, and finally gives the correct value, $r1 = sum(0, n/2)$, however that's all thread 1 does. So, we end in a state where $r \neq sum(0, 26)$.

(b) Expected number of times to find the bug in testing is $(1.3 * 10^{14})/78 \approx 1.66 * 10^{12}$. This means, if you run the program 1.6 trillion times you are expected to encounter one buggy execution path. Having a script running 1000 times per second, it will still take you 1.66 billion seconds, or 19,290 days, or 52.85 years to find the bug.

(c) $(1.66 * 10^{12})/(10^{11}) \approx 16.66$ days. It is expected to take about 17 days for a user to discover the bug.

(d) Testing is not the best way make sure a program does what it's intended and doesn't have any bugs, as indicated by the above answers. It can take days, months, or even more for testing to find a bug. By the time your tests find the bug, or enough users finds and reports the bug, it will already have been too late and the damage has been done. For a mission critical system, testing is not enough and program verification is a must. As such, it is a good thing that I took CS 536 and learned about program verification.

Task 5

I spent about 5 hours on this.