**Department of Computer Science** BOSTON UNIVERSITY

# CS-350 - Fundamentals of Computing Systems

Midterm Exam #2

Fall 2020

Name: Jae Hong Lee

Login: jhonglee@bu.edu BU ID: U27565203

**Submission instructions:** You can either (i) take a print out of the booklet, write your answers down on it, and then scan it as PDF before submitting to Gradescope, or (ii) use the Latex template to to type in answers and submit the PDF through Gradescope, or (iii) simply copy the questions to your favorite text editor, type in your answers below every question, and upload a PDF through Gradescope. **NOTE:** *when uploading your submission, make sure to associate your answers with the correct PDF page.*

| | |
|---|---|
| Problem #1: | /17 |
| Problem #2: | /25 |
| Problem #3: | /26 |
| Problem #4: | /24 |
| Problem #5: | /18 |
| Total Score: | /110 |

**Note:**

- This is a open-book/notes exam.
- Cite any external source you use.
- Calculators are allowed.
- You have 24 hours to complete your exam.
- There are 110 total points.
- Show your work for full marks.
- **Explain all your assumptions clearly.**

I give credit to students who can accurately assess their performance. Thus, after you finish the exam, answer the following for up to five bonus points!

Bonus = max(0, 5 - |GuessedGrade - ActualGrade|)

**What do you guess your score (out of 110) will be?** ____________

# Problem 1

The iPhony XX is the latest and greatest folding phone that the year 2020 has gifted us. The new feature that has everyone talking is the *clam-lock* function. That is, upon detecting the face of a stranger trying to use the device, the phone folds itself in half and physically locks. Just like a clam. Hence the name. While the gadget has been extensively covered in a multitude of unboxing videos, it is unclear if the clam-lock is effective. That's because if it takes too long—more than 5 seconds—to detect a stranger, initiate and complete locking, then the security of the device can be compromised. As an editor of the ComputahReports magazine, you have decided to conduct a series of measurements to determine the responsiveness of the clam-lock.

**(a) [5 points]** According to the few available statistics, it appears that the variance on the time it takes for the clam-lock to perform detection&locking is 1.44. How many times would you need to measure the behavior of the clam-lock if the goal is to construct a 96.6% confidence interval on the detect&lock time with an error bound of $\pm 0.3$ seconds?

Variance 1.44

**(b) [6 points]** You decide to acquire 100 samples of the behavior of the acclaimed feature. As it turned out, the variance was very much spot on! Also, from your experiments, you have obtained the following 96.6% confidence interval on the clam-lock timing: $[4.49s, 5.11s]$. The problem is that the upper-end of the interval is beyond the 5 seconds mark and it would not look good on the review article. We want to tighten the error bound so that the upper-end is exactly $5s$ while dropping our confidence to 95%. Do we need to acquire more samples to pull that off? If so, how many more? (*Hint: you can assume that, as you acquire more samples, average and variance stay the same*).

**(c) [6 points]** Regardless of what computed above, you decide to acquire 20 more samples. The result of these additional samples is reported in Table 1. For simplicity, assume that the variance over the entire 120 samples has not changed. Consider however that the average might change with the new samples. What is the 95% confidence interval for the clam-lock timing?

Table 1: Additional 20 samples of clam-lock timing.

| Sample # | Value (s) | Sample # | Value (s) | Sample # | Value (s) | Sample # | Value (s) |
|---|---|---|---|---|---|---|---|
| 1 | 5.1 | 6 | 5.1 | 11 | 4.7 | 16 | 5 |
| 2 | 4.4 | 7 | 4.9 | 12 | 4.5 | 17 | 4.4 |
| 3 | 5 | 8 | 5.2 | 13 | 4.5 | 18 | 4.9 |
| 4 | 5.1 | 9 | 5 | 14 | 4.3 | 19 | 4.4 |
| 5 | 4.6 | 10 | 4.6 | 15 | 4.7 | 20 | 4.4 |

# Problem 2

The cloud enterprise you are working for offers database solutions for analytics on big data. The enterprise has thousands of machines but they pre-allocate customers to machines so that it is easier to manage the workload. On each machine, the problem boils down to the following problem. The database has three very large data tables, namely D1, D2, D3 that are stored in a network-accessible storage unit (NAS). Queries that arrive at the machine only use two of them. And the machine has only enough memory to hold only two tables at any point in time, but not all three of them.

For a query to execute, the two tables required by the query need to be in memory on the local machine. If the table is not already in memory at the time a query is scheduled for execution, it needs to be loaded from the NAS. The time it takes to load each of the D1-D3 tables is reported in Table 2. Removing a table from memory does not take any time since tables are simply overwritten.

Table 2: Time to load database tables from NAS.

| Table # | Load Time (s) |
|---|---|
| D1 | 3 |
| D2 | 2 |
| D3 | 1 |

At time 0, no table is in memory. Table 3 lists a sequence of queries arriving at a machine under analysis, their arrival time, execution time (assuming that the required tables are in memory), and the data tables required to execute them.

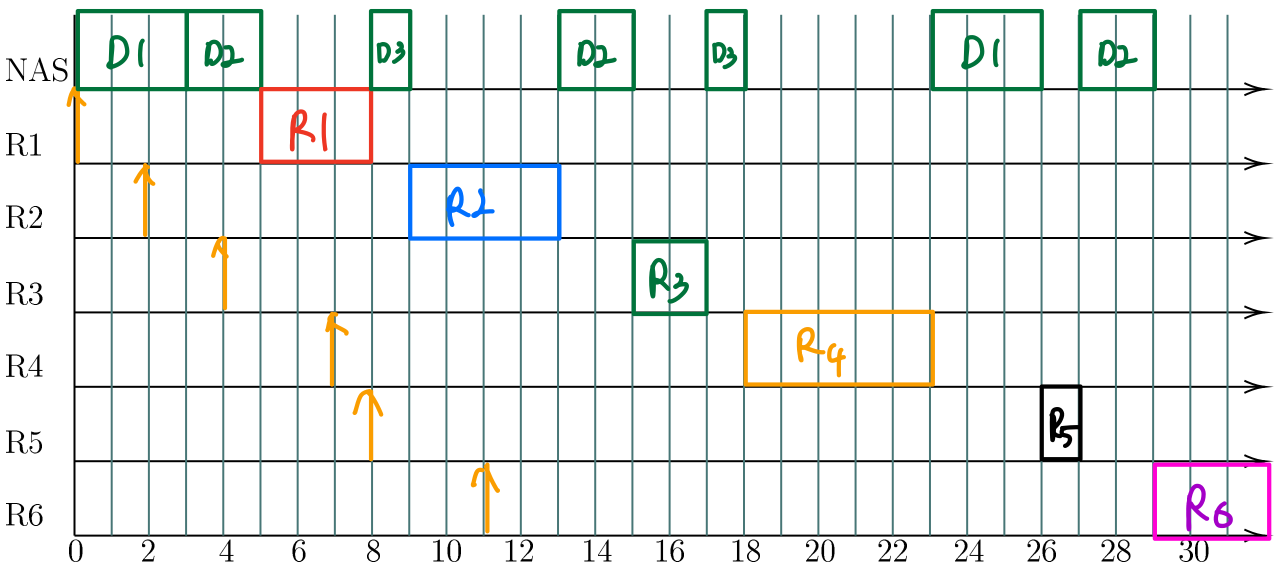
Table 3: Arrival sequence of queries.

| Request ID # | Arrival Time (s) | Execution Time (s) | Required Tables |
|---|---|---|---|
| R1 | 0 | 3 | D1, D2 |
| R2 | 2 | 4 | D1, D3 |
| R3 | 4 | 2 | D1, D2 |
| R4 | 7 | 5 | D2, D3 |
| R5 | 8 | 1 | D1, D3 |
| R6 | 11 | 3 | D2, D3 |

9

(a) **[5 points]** Currently, the company has adopted a FIFO strategy to schedule query execution on all the machines. Draw the schedule produced by the scheduler on the workload provided in Table 3. In the schedule, highlight which query is scheduled at each time instant and the time spent loading tables.




(b) **[2 points]** A colleague suggests that one could achieve better fairness in the system by employing Round Robin instead. Is this a good idea? Provide a brief reasoning to motivate your answer.

Round Robin means for setted time quantum ex) 1 t switch between request. but in this system processor needs to load data each time swiching request, the fairness could be decreased.

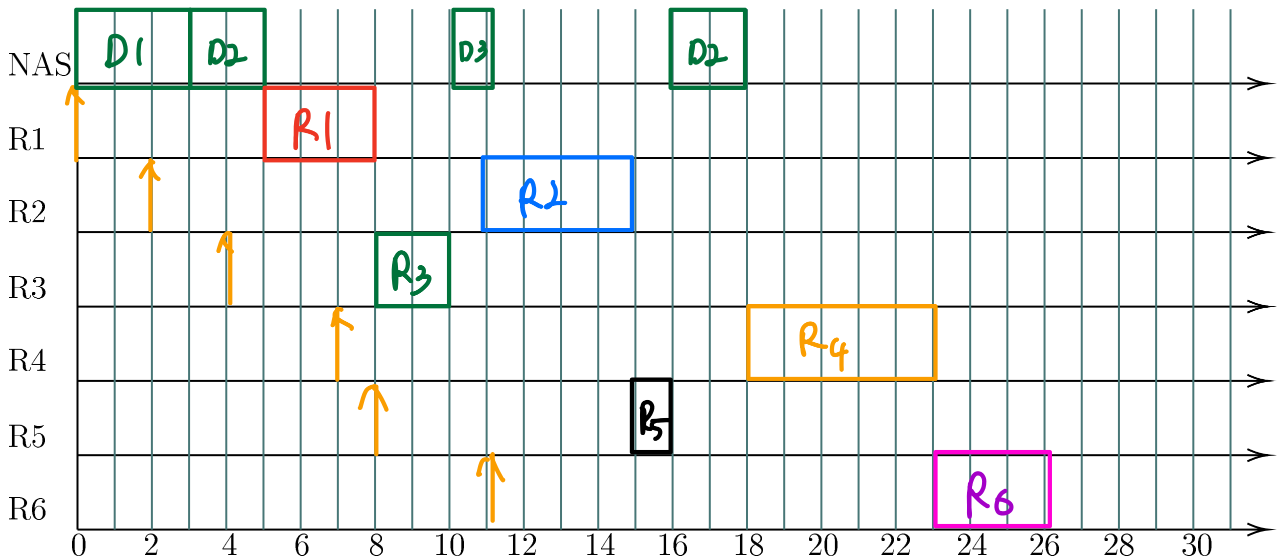
 

D2 D3 D1 D2



**(c) [8 points]** Devise a scheduling strategy to minimize the average response time of the considered requests. Provide a description of your new policy and draw the resulting schedule.

Not preemptive.




Shortest data load first.

First Read, First come, first send!

R1 D1 D2 ✓

R2 D1 D3 ✓ 1

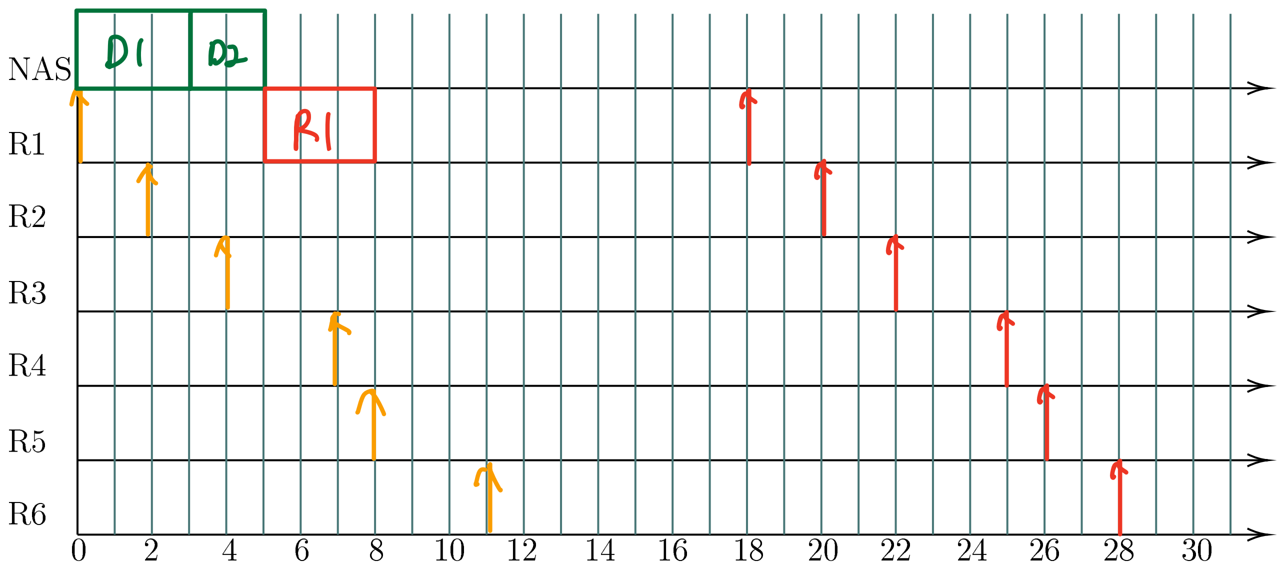
R3 D1 D2 ✓

R4 D2 D3 1

R5 D1 D3 1

R6 D2 D3 1

(d) **[8 points]** You have deployed your policy and it works very well. However, the expectation that users have from the system has now increased. Indeed, some users have started to complain that some queries are being returned too slowly. Specifically, they mention that having a response time of 18 seconds or more is just unacceptable! Patch the strategy you devised above to try to meet the constraint on the response time. The new strategy does not have to *always* meet the requirement, but it should be able to with the workload in Table 3. Provide a description of your new policy and draw the resulting schedule.




(e) **[2 points]** At time 32, a new query arrives with an unknown execution time. If the system uses a sliding window average with window size 3, what is the predicted execution time for the new query? For simplicity, assume that the queries were executed in FIFO order.

3

$\frac{5+1+3}{3} = 3$

## Problem 3

You are trying to assemble your very first racing QuadRotor using a small single-core micro-controller. For the quadrotor to be stable and reliable while performing aerobatic maneuvers the micro will needs to execute a set of sense, control, and actuate tasks with strict temporal constraints. Specifically, sampling from the inertial measurement unit (IMU) needs to be performed at 100 Hz, with each IMU sampling instance taking at most 3 ms. Next, a Proportional+Integrative+Derivative (PID) controller needs to be executed every 18 ms, with any PID job taking between 4 and 6 ms. Lastly, actuation commands (ACT) need to be periodically sent to the motors which in the worst-case takes 9 ms and needs to be performed every 34 ms.

(a) **[4 points]** In a first revision of this project, you decide to try and see if implementing an RM scheduler in your micro-controller can be sufficient. Is RM capable of handling the workload? Motivate your answer.

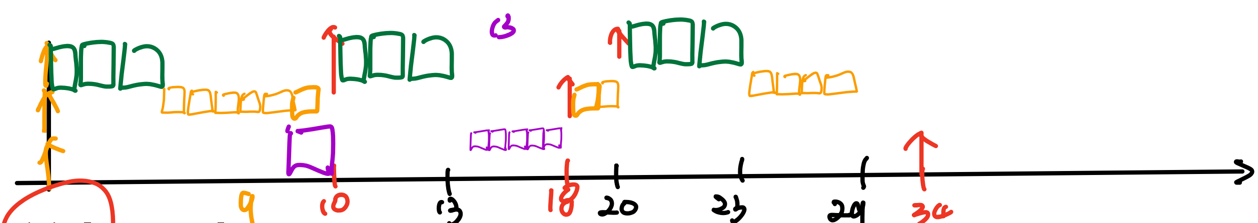
Schedulable

| | WCET | Ti | Utilization |
|---|---|---|---|
| IMU | 3ms | 10ms | 0.3 |
| PID | 6ms | 18ms | 0.3333 |
| ACT | 9ms | 34ms | 0.2647 |

$3(2^{\frac{1}{3}}-1) = 0.779.$

inconclusive

0.8980 . >,




(b) **[2 points]** Before finalizing the design, you realize that you will need to add an additional task to receive commands form the radio transmitter (RDT). The receiver task needs to operate at 50 Hz, with each *receive* operation taking up to 5.4 ms to complete. Is the current design still feasible with RM and the considered micro-processor?

| | WCET | Ti | Utilization |
|---|---|---|---|
| IMU | 3ms | 10ms | 0.3 |
| PID | 6ms | 18ms | 0.3333 |
| ACT | 9ms | 34ms | 0.2647 |
| RDT= | 5.4ms | 20ms | 0.27 |

$\}$ 1.16

$\boxed{\text{No}}$. If possible ?

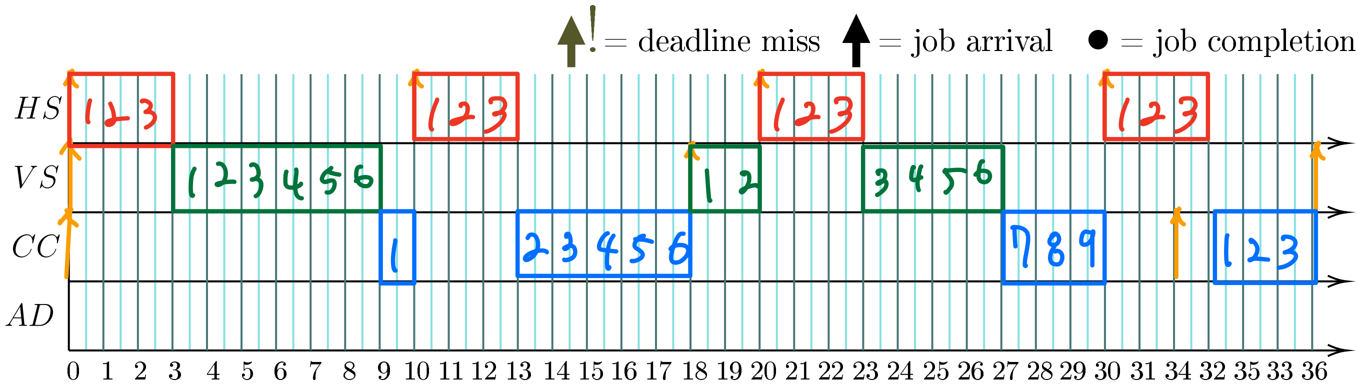
 

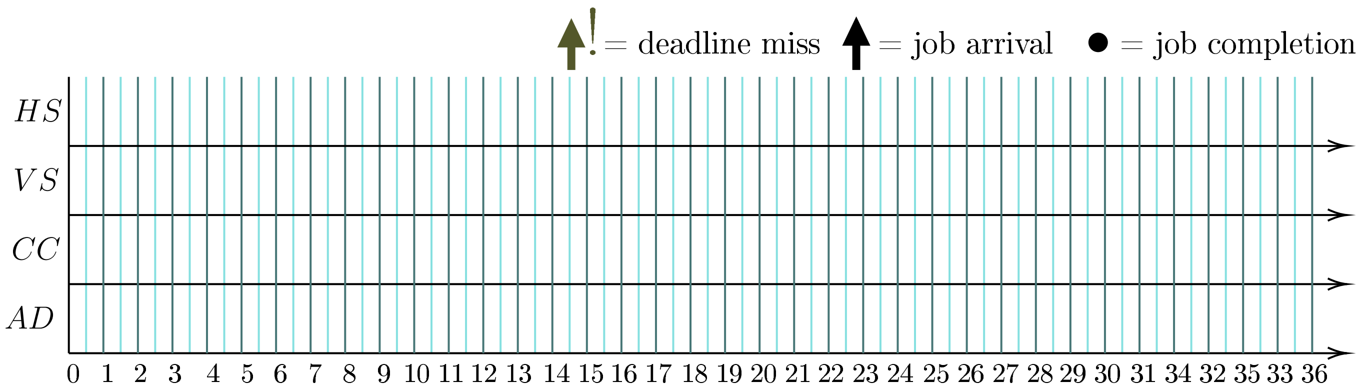
WCET Ti Utilization
IMU 3ms 10ms 0.3

PID 6mS 18mS 0.3333
ACT 9mS 34mS 0.2647



**[EXTRA GRIDS (*Use only if necessary and reference properly.*)]**




| Step | Thread T1 | Thread T2 |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

(c) **[5 points]** During a re-design of your platform, you come across a very similar micro-processor in terms of form factor, pinout and cost. The only difference is that the new micro has 2 cores, but a clock speed that is only 60% of the previous single-core micro. By considering the tasks ordered in the same way they appear in this problem, determine is RM-FF can be used to schedule the taskset.

Not schedulable.

| | WCET | $T_i$ | Utilization |
|---|---|---|---|
| IMU | 5ms | 10ms | 0.5 |
| PID | 10ms | 18ms | 0.5555 |
| ACT | 15ms | 34ms | 0.441 |
| ROT= | 9 ms | 20ms | 0.45 |

$\} \; 1.946$

$2(\sqrt{2}-1)$

$\underline{0.828}$

| P1 | P2 |
|---|---|
| ACT<br>IMU | X<br>PID |

(d) **[5 points]** By considering the same ordering above, determine if the taskset can be schedulable using EDF-FF.

Sched

Not Schedun.

EDF $\frac{\beta \cdot 2+1}{\beta+1} = \frac{3}{2} = \underline{1.5}$

$\left\lfloor \frac{1}{\max C_i/T_i} \right\rfloor = 1$

**(e)** **[5 points]** Is the taskset at all schedulable by any partitioned scheduling strategy that uses RM or EDF on each core? Motivate your answer.

**(f)** **[5 points]** Is the taskset schedulable under Global EDF?

# Problem 4

In a town called Trecolori in Georgia, people are politically very involved. The city is governed by three political parties, the Blue, the Red, and the Green party. Each party has a known number of representatives: $KB$, $KR$, and $KG$, respectively. Because the three parties do not trust each other to handle the finances of the town, the central bank of Trecolori has a vault that can only be opened when three keys are simultaneously used—one per party.

All the representatives for the Blue party have the same blue key, and the same is true for the Red and Green party. So it is enough that, when it's time to open the vault, at least one representative of each party is present to unlock the corresponding part of the lock.

Only the City Treasurer—who does not have any official party affiliation—is allowed inside the vault. So when the vault is unlocked, only the treasurer is allowed to open the door of the vault, enter to perform its daily duty, and then close the door of the vault.

After the treasurer is done with the vault, it is responsibility of the same representatives that unlocked the vault to lock it back before they can leave.

Every day, when the bell tower of the city marks noon, it's time for the treasurer to open the vault. All the representatives are notified and all of them will try to get to the bank to unlock (their part of) the vault. If another representative of the same party has already arrived to help with the vault, there is nothing to do until the next day.

**NOTE**: it can happen that if representatives are too slow in their progress, opening the vault is skipped on a given day. But it should never happen that the vault is opened twice in the same day.

**(a) [12 points]** Complete the pseudo-code reported below with appropriate statements. You should have at most one statement per blank; not all the blanks need to be used; you should not need to use more statements than the number of blanks. Any loop construct (e.g. `for...loop`, `while...loop`) takes two statements; an `if...else...endif` takes three statements, and so on. If the same variable is accessed by multiple processes, the variable is shared.

```
Process BellTower:
    while (true) { /* Repeat Forever */
       wait_until_noon(); /* Returns if noon passed */
       for (i = 0; i < KB + KR + KG; ++i) {
          Signal (wakey)
          __________________
       } loop;

       /* ??? */
       for (i = 0; i < KB + KR + KG; ++i) {
          wait(bank_done);
       } loop;

       wait(vault_done);
    } loop;
```

Listing 1: Pseudo-code of Bell Tower Process

```
Process Treasurer:
    while (true) { /* Repeat Forever */
       for (j in [B, R, G]) {
          wait(vault_unlock[j]);
          __________________
       } loop;

       open_and_enter_vault();
       do_stuff_in_vault();
       exit_and_close_vault();

       Signal (valut_done);

    } loop;
```

Listing 2: Pseudo-code of Treasurer

```
Process C_Representative:
    while (true) { /* Repeat Forever */
       wait(wakey);

       go_to_bank();

       wait(m)
       arrived[C] = 1;
       if (arrived[C] == 1) {
          ------------------
          arrived[C] = KG;        G, 1
       } else {
          if (arrived[C] == KC) {
             arrived[C] = 0;
          } endif;
          ------------------
          wait( arrived[C]);
          ------------------
          ------------------
          continue;
       } endif;

       go_to_vault();

       /* Turn the key of color C */
       turn_partial_key(C);

       signal(vault_unlocked[C]);

       wait(vault_done);
       signal(vault_done);

       /* Un-turn and remove key of color C */
       unturn_partial_key(C);
       leave_bank();
       ------------------
       ------------------
       ------------------
       ------------------

       signal(bank_done);
    } loop;
```

Listing 3: ]Pseudo-code of Representative of Color C in [B, R, G]

**(b) [5 points]** Provide an initialization for all the shared variables used in the listings above. These are: the array of integers `arrived`, the array of semaphores `vault_unlocked`, the semaphores `wakey`, `bank_done`, `vault_done`, as well as any variable that you have added to complete the code (if any).

**(c) [7 points]** Look at the BellTower process in Listing 1. What is the purpose of the for-loop that is marked with the "???" comment? What happens to the correctness of the code if the entire loop (and whatever inside it) is removed? Briefly motivate your answer.

# Problem 5

Consider a system with 4 resources $R_1, R_2, R_3$, and $R_4$. In the system there are 5 processes $P_1$ through $P_5$ that compete for access to the resources. To perform deadlock avoidance, the system employs the Banker Algorithm. The immutable system parameters are given in Table 4, while the (incomplete!) state of the system at a certain point in time is given in Table 5.

| | | Parameter | Resources | | | |
|---|---|---|---|---|---|---|
| | | | $R_1$ | $R_2$ | $R_3$ | $R_4$ |
| | | $R(k)$ | 3 | 17 | 16 | 12 |
| Processes | $P_1$ | $C_1(k)$ | 0 | 2 | 1 | 0 |
| | $P_2$ | $C_2(k)$ | 1 | 6 | 5 | 2 |
| | $P_3$ | $C_3(k)$ | 2 | 3 | 6 | 6 |
| | $P_4$ | $C_4(k)$ | 0 | 6 | 5 | 2 |
| | $P_5$ | $C_5(k)$ | 0 | 6 | 5 | 6 |

Table 4: Static parameters for the considered system.

| | | Parameter | Resources | | | | Parameter | Resources | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $R_1$ | $R_2$ | $R_3$ | $R_4$ | | $R_1$ | $R_2$ | $R_3$ | $R_4$ |
| | | $V(k)$ | | | | | | | | | |
| Processes | $P_1$ | $A_1(k)$ | 0 | 1 | 1 | 0 | $N_1(k)$ | | | | |
| | $P_2$ | $A_2(k)$ | 1 | 2 | 3 | 1 | $N_2(k)$ | | | | |
| | $P_3$ | $A_3(k)$ | 1 | 3 | 6 | 5 | $N_3(k)$ | | | | |
| | $P_4$ | $A_4(k)$ | 0 | 6 | 3 | 2 | $N_4(k)$ | | | | |
| | $P_5$ | $A_5(k)$ | 0 | 0 | 1 | 4 | $N_5(k)$ | | | | |

Table 5: System state for considered system.

**(a)** **[4 points]** Complete any missing parameter in Table 5.

**(b)** **[7 points]** Determine if the current state, as reported in Table 5, is safe. Show your reasoning.

**(c) [7 points]** If the system was determined to be in a safe state, can we grant a request coming from $P_2$ to allocate 2 additional units of $R_2$ and 1 additional unit of $R_3$? Show your work.