DESEMBRANA, Anna Patricia B. S13
MANDADERO, Clarissa Mae S. S13

Project 2 – Synchronization Problems

Synchronization Technique

We used Semaphore as the synchronization technique for this project, more specifically a counting semaphore. We decided to use this technique since the problem allows multiple instances of the same resource to access the critical section and a solution for this involves the usage of counting semaphores.

List of Variables

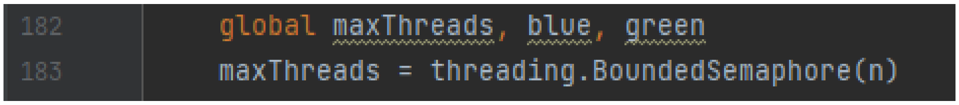
Variable Name	**Data Type**	**Purpose**
n	int	Number of slots inside the fitting room
b	int	Number of blue threads
g	int	Number of green threads
nInside	int	Number of threads inside the fitting room
maxThreads	BoundedSemaphore	Counting semaphore used to limit the number of threads inside the fitting room
currColor	string	Indicates the current color of threads in the fitting room
bReady	Boolean list	Used to check if a blue thread with a specific ID number is ready to enter the critical section. The value is set to “True” if it is ready.
gReady	Boolean list	Used to check if a green thread with a specific ID number is ready to enter the critical section. The value is set to “True” if it is ready.
bCounter	int	Refers to the index of the last blue thread that entered the fitting room in one iteration
gCounter	int	Refers to the index of the last green thread that entered the fitting room in one iteration
blue_threads	list	Contains blue colored_threads
green_threads	list	Contains green colored _hreads

Satisfaction of Constraints

I. Constraint A

Condition:
There are only n slots inside the fitting room of a department store. Thus, there can only be at most n persons inside the fitting room at a time.

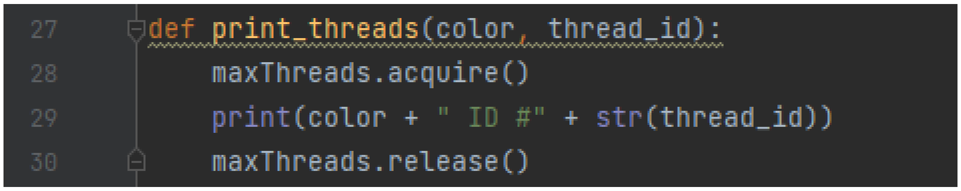
Satisfaction of Constraint:
We used a counting semaphore to ensure that there is a limit on the number of threads that can enter the critical condition at the same time. A BoundedSemaphore was created with variable name `maxThreads` whose limit is set to the `n` given by the user. See images of the code below to see its usage:



```
    global maxThreads, blue, green
    maxThreads = threading.BoundedSemaphore(n)
```

Figure A.1

Figure A.1 shows the initialization of `maxThreads`



```
def print_threads(color, thread_id):
    maxThreads.acquire()
    print(color + " ID #" + str(thread_id))
    maxThreads.release()
```

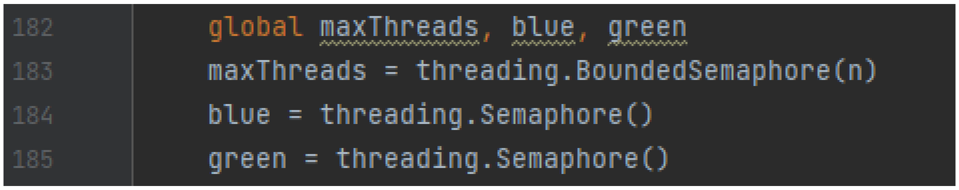
Figure A.2

Figure A.2 shows how `maxThreads` was used to ensure that only n number of persons can fit inside the fitting room

II. Constraint B

Condition:
There cannot be a mix of blue and green in the fitting room at the same time. Thus, there can only be at most n blue threads or at most n green threads inside the fitting room at a time.

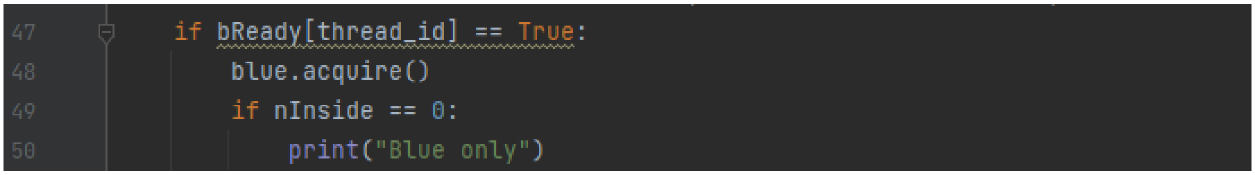
Satisfaction of Constraint:
We used a binary semaphore for locking the execution of blue or green threads. Semaphores for each color, `green` and `blue`, were created, and they were each used in their corresponding methods for the execution of threads. See the image of the code below to see its usage:



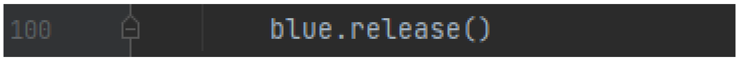
```
global maxThreads, blue, green
maxThreads = threading.BoundedSemaphore(n)
blue = threading.Semaphore()
green = threading.Semaphore()
```

Figure B.1

Figure B.1 shows the initialization of the binary semaphores `blue` and `green`



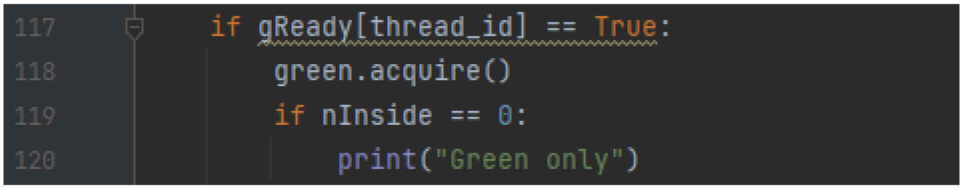
```
if bReady[thread_id] == True:
    blue.acquire()
    if nInside == 0:
        print("Blue only")
```



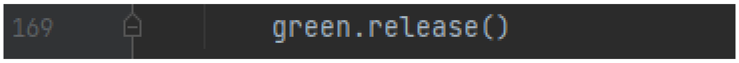
```
        blue.release()
```

Figures B.2 and B.3

Figure B.2 locks the blue thread after checking if the thread with the specific ID number is ready to execute and releases it after execution as shown in Figure B.3



```
if gReady[thread_id] == True:
    green.acquire()
    if nInside == 0:
        print("Green only")
```



```
        green.release()
```

Figures B.4 and B.5

Figure B.4 locks the blue thread after checking if the thread with the specific ID number is ready to execute and releases it after execution as shown in Figure B.5

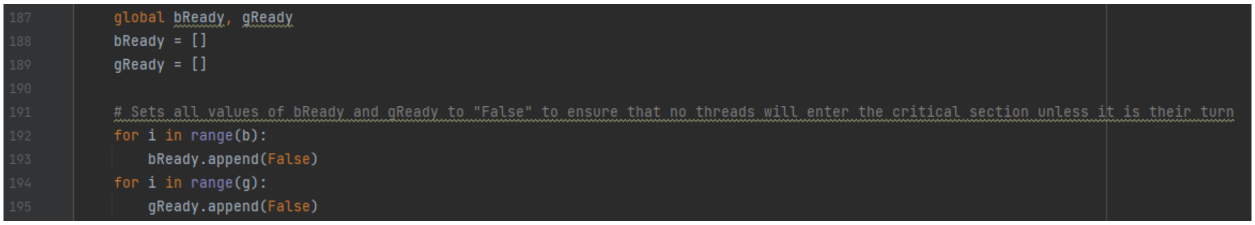
III. Constraint C

Condition:

The solution should not result in deadlock.

Satisfaction of Constraint:

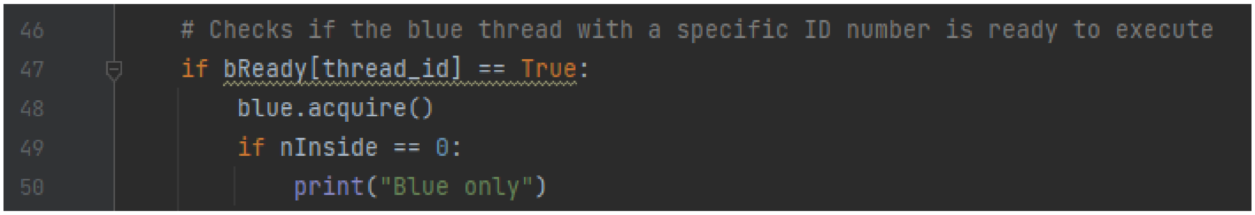
We used `bReady[]` and `gReady[]` to ensure that only the threads that are ready will be allowed to enter the fitting room or critical section. If the value of `bReady[]` or `gReady[]` is set to "True" then that means that specific thread can now enter the critical section. If "False", then it means that it has to wait for its turn. This is checked before letting the threads enter the critical section as seen in the snippet of the code below:



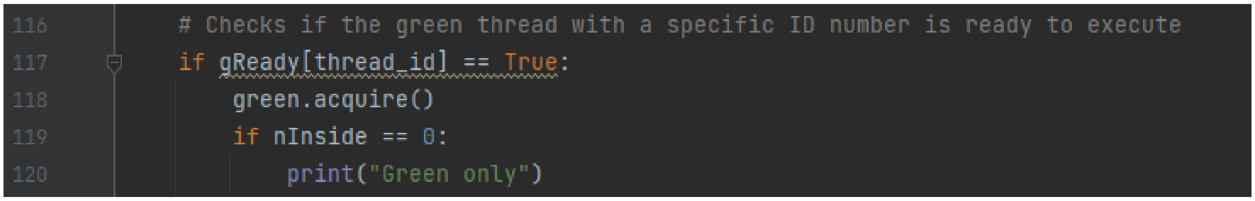
```
global bReady, gReady
bReady = []
gReady = []

# Sets all values of bReady and gReady to "False" to ensure that no threads will enter the critical section unless it is their turn
for i in range(b):
    bReady.append(False)
for i in range(g):
    gReady.append(False)
```

Figure C.1

Figure C.1 shows the initialization of `bReady[]` and `gReady[]`. All of its values are set to “False” first to ensure that no threads will enter the fitting room unless it is their turn to enter.



```
# Checks if the blue thread with a specific ID number is ready to execute
if bReady[thread_id] == True:
    blue.acquire()
    if nInside == 0:
        print("Blue only")
```



```
# Checks if the green thread with a specific ID number is ready to execute
if gReady[thread_id] == True:
    green.acquire()
    if nInside == 0:
        print("Green only")
```

Figures C.2 and C.3

Figures C.2 and C.3 use `bReady[]` and `gReady[]` with the `thread_id` as the index to check if the corresponding thread is ready to enter the fitting room.

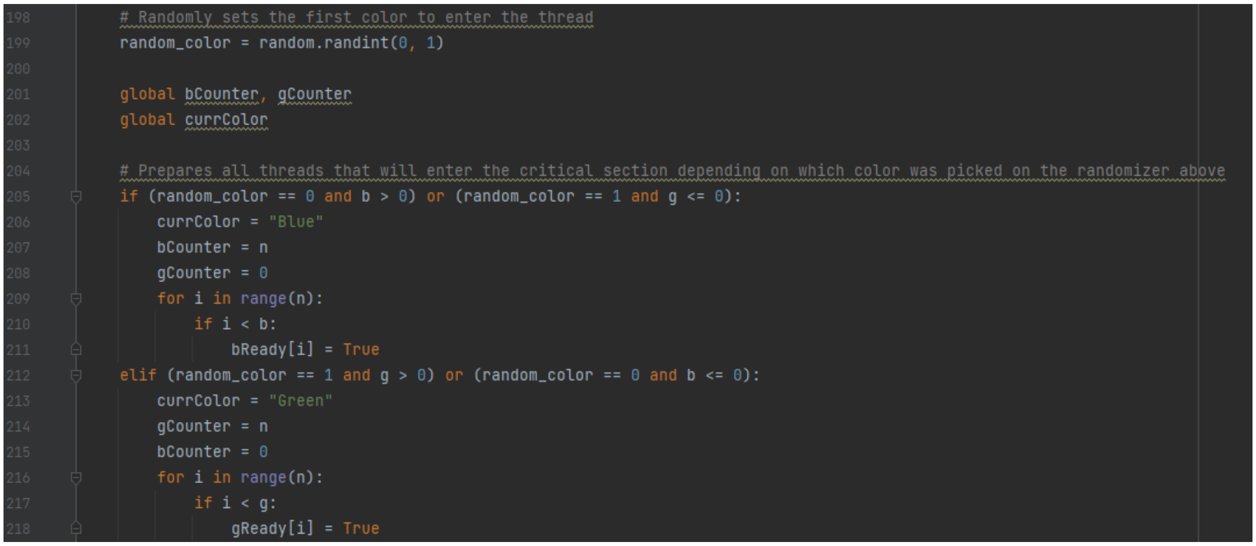
IV. Constraint D

Condition:

The solution should not result in starvation. For example, blue threads cannot forever be blocked from entering the fitting room if there are green threads lining up to enter as well.

Satisfaction of Constraint:

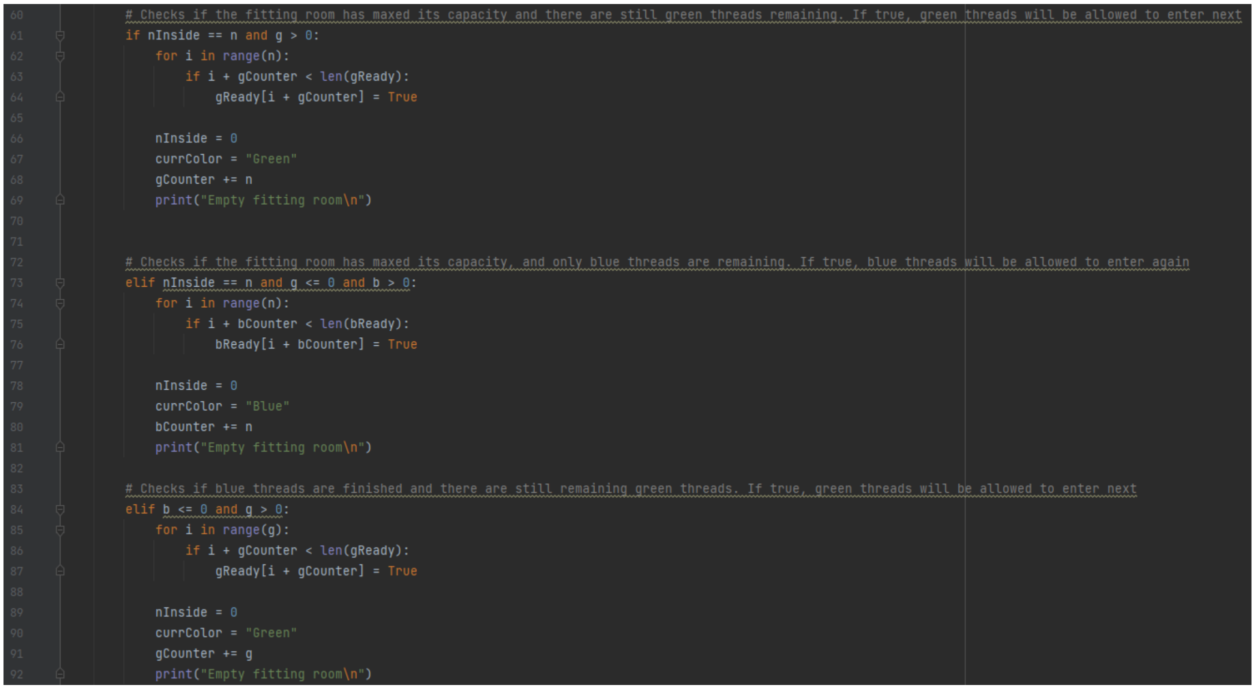
We programmed the threads to enter the fitting room alternately to ensure that it will not result in starvation. In case one of the threads has finished processing, the other color’s remaining threads will be processed repeatedly until it finishes.



```python
    # Randomly sets the first color to enter the thread
    random_color = random.randint(0, 1)

    global bCounter, gCounter
    global currColor

    # Prepares all threads that will enter the critical section depending on which color was picked on the randomizer above
    if (random_color == 0 and b > 0) or (random_color == 1 and g <= 0):
        currColor = "Blue"
        bCounter = n
        gCounter = 0
        for i in range(n):
            if i < b:
                bReady[i] = True
    elif (random_color == 1 and g > 0) or (random_color == 0 and b <= 0):
        currColor = "Green"
        gCounter = n
        bCounter = 0
        for i in range(n):
            if i < g:
                gReady[i] = True
```

Figure D.1

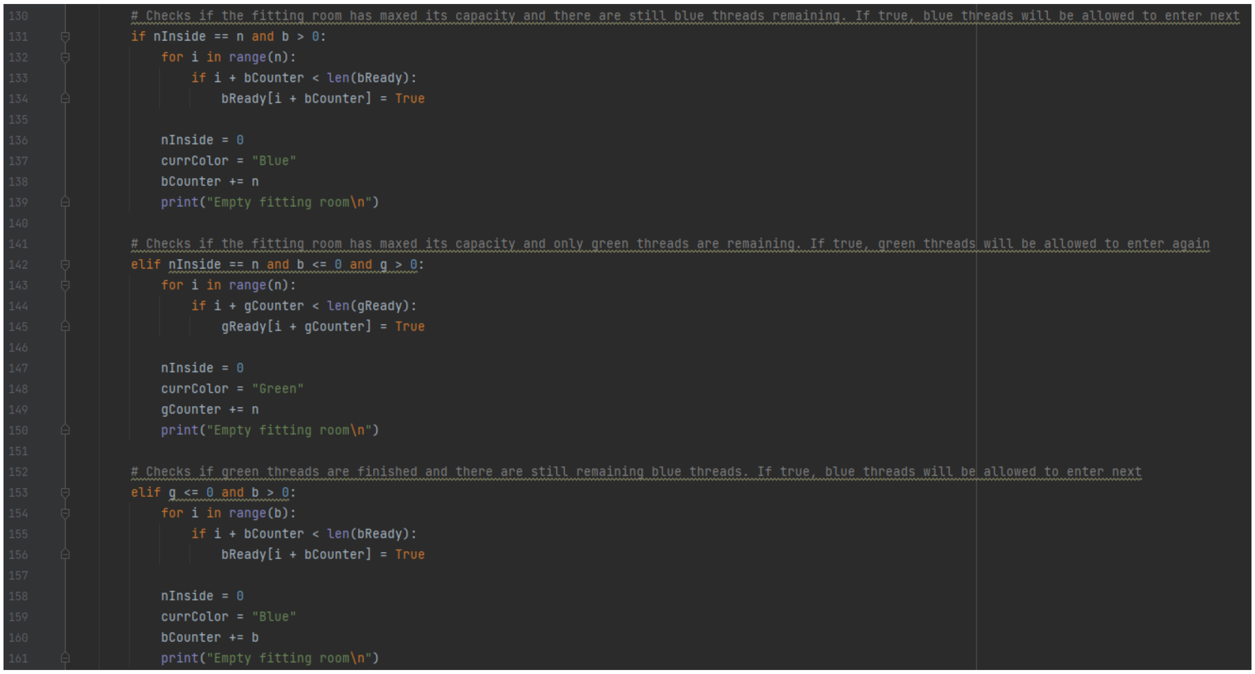
Figure D.1 shows the process in choosing the first color to enter the fitting room. It is chosen through a randomizer. Once chosen, the if-else condition will set that first color's `n` threads to ready so that it can start entering the fitting room.



```python
        # Checks if the fitting room has maxed its capacity and there are still green threads remaining. If true, green threads will be allowed to enter next
        if nInside == n and g > 0:
            for i in range(n):
                if i + gCounter < len(gReady):
                    gReady[i + gCounter] = True

            nInside = 0
            currColor = "Green"
            gCounter += n
            print("Empty fitting room\n")


        # Checks if the fitting room has maxed its capacity, and only blue threads are remaining. If true, blue threads will be allowed to enter again
        elif nInside == n and g <= 0 and b > 0:
            for i in range(n):
                if i + bCounter < len(bReady):
                    bReady[i + bCounter] = True

            nInside = 0
            currColor = "Blue"
            bCounter += n
            print("Empty fitting room\n")

        # Checks if blue threads are finished and there are still remaining green threads. If true, green threads will be allowed to enter next
        elif b <= 0 and g > 0:
            for i in range(g):
                if i + gCounter < len(gReady):
                    gReady[i + gCounter] = True

            nInside = 0
            currColor = "Green"
            gCounter += g
            print("Empty fitting room\n")
```

Figure D.2

The if-else conditions found in figure D.2 checks if blue has finished processing its threads, thus signaling that the fitting room is now empty. Once the fitting room becomes empty, the color that is allowed to enter the fitting room now changes to green, if green still has remaining threads. If green threads have finished processing, the blue threads will be asked to enter the fitting room again until all threads have been processed.



```python
        # Checks if the fitting room has maxed its capacity and there are still blue threads remaining. If true, blue threads will be allowed to enter next
        if nInside == n and b > 0:
            for i in range(n):
                if i + bCounter < len(bReady):
                    bReady[i + bCounter] = True

            nInside = 0
            currColor = "Blue"
            bCounter += n
            print("Empty fitting room\n")

        # Checks if the fitting room has maxed its capacity and only green threads are remaining. If true, green threads will be allowed to enter again
        elif nInside == n and b <= 0 and g > 0:
            for i in range(n):
                if i + gCounter < len(gReady):
                    gReady[i + gCounter] = True

            nInside = 0
            currColor = "Green"
            gCounter += n
            print("Empty fitting room\n")

        # Checks if green threads are finished and there are still remaining blue threads. If true, blue threads will be allowed to enter next
        elif g <= 0 and b > 0:
            for i in range(b):
                if i + bCounter < len(bReady):
                    bReady[i + bCounter] = True

            nInside = 0
            currColor = "Blue"
            bCounter += b
            print("Empty fitting room\n")
```

Figure D.3

The if-else conditions found in figure D.3 checks if green has finished processing its threads, thus signaling that the fitting room is now empty. Once the fitting room becomes empty, the color that is allowed to enter the fitting room now changes to blue, if blue still has remaining threads. If blue threads have finished processing, the green threads will be asked to enter the fitting room again until all threads have been processed.