Exercise 1)

```
445 void StartDefaultTask(void const * argument)
446 {
447   /* USER CODE BEGIN 5 */
448
449   /* Infinite loop */
450   for(;;)
451   {
452       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);//Enable RED LED
453       osDelay(2000); //Delay 2ms
454       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);//Disable RED LED
455       osDelay(500);// 0.5ms delay
456   }
457   /* USER CODE END 5 */
458 }
```

Exercise 2)

```
463       TickType_t TaskTimeStamp; //create variable of type TickType_t
464       TickType_t DelayTimeMsec = 2000; //create variable of type TickType_t
465       TaskTimeStamp = xTaskGetTickCount(); //load variable with current starting tick value
466
467       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);//Enable Green LED
468       osDelayUntil(&TaskTimeStamp, DelayTimeMsec); //Delay 2 sec . Function creates delay relative to specific point in time
469       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);//Disable Green LED
470       osDelayUntil(&TaskTimeStamp, DelayTimeMsec);
```

Exercise 3)

```
451 void StartFlashGreenLedTask(void const * argument)
452 {
453   /* USER CODE BEGIN 5 */
454     TickType_t TaskTimeStamp;
455     TickType_t DelayTimeMsec = 2000;// 2 seconds
456   /* Infinite loop */
457   for(;;)
458   {
459       //Exercise 1
460       /*
461       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);//Enable RED LED
462       osDelay(2000); //Delay 2ms
463       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);//Disable RED LED
464       osDelay(500);// 0.5ms delay
465       */
466
467       // Exercise 2
468       /*
469       TickType_t TaskTimeStamp; //create variable of type TickType_t
470       TickType_t DelayTimeMsec = 2000; //create variable of type TickType_t
471       TaskTimeStamp = xTaskGetTickCount(); //load variable with current starting tick value
472
473       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);//Enable Green LED
474       osDelayUntil(&TaskTimeStamp, DelayTimeMsec); //Delay 2 sec . Function creates delay relative to specific point in time
475       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);//Disable Green LED
476       osDelayUntil(&TaskTimeStamp, DelayTimeMsec);
477       */
478
479       // Exercise 3
480       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);//Enable Green LED
481       osDelayUntil(&TaskTimeStamp, DelayTimeMsec); //Delay 2 sec . Function creates delay relative to specific point in time
482       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);//Disable Green LED
483       osDelayUntil(&TaskTimeStamp, 1500);//1.5 sec
484       //osDelay(1);
485
486   }
487   /* USER CODE END 5 */
488 }
```

```
497 void StartFlashRedLedTask(void const * argument)
498 {
499   /* USER CODE BEGIN StartFlashRedLedTask */
500     TickType_t TaskTimeStamp;
501     TickType_t DelayTimeMsec = 1000;
502   /* Infinite loop */
503   for(;;)
504   {
505       // Exercise 3
506       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);//Enable RED LED
507       osDelayUntil(&TaskTimeStamp, DelayTimeMsec); //Delay 2 sec . Function creates delay relative to specific point in time
508       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);//Disable Red LED
509       osDelayUntil(&TaskTimeStamp, DelayTimeMsec);
510       //osDelay(1);
511   }
512   /* USER CODE END StartFlashRedLedTask */
513 }
```

Exercise 4)

```
130   /* Create the thread(s) */
131   /* definition and creation of FlashGreenLedTa */
132   osThreadDef(FlashGreenLedTa, StartFlashGreenLedTask, osPriorityAboveNormal, 0, 128);
133   FlashGreenLedTaHandle = osThreadCreate(osThread(FlashGreenLedTa), NULL);
134
135   /* definition and creation of FlashRedLedTask */
136   osThreadDef(FlashRedLedTask, StartFlashRedLedTask, osPriorityNormal, 0, 128);
137   FlashRedLedTaskHandle = osThreadCreate(osThread(FlashRedLedTask), NULL);
```

```
488       // Exercise 4
489       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,GPIO_PIN_SET);//enable Blue LED
490
491       for(int i=0; i<=160;i++)
492       {
493           HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);//enable Green LED
494           osDelay(25);
495       }
496       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,GPIO_PIN_RESET);//disable Blue LED
497       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_RESET);//disable Green LED
498       osDelay(6000);
```

```
511 void StartFlashRedLedTask(void const * argument)
512 {
513   /* USER CODE BEGIN StartFlashRedLedTask */
514     TickType_t TaskTimeStamp;
515     TickType_t DelayTimeMsec = 1000;
516   /* Infinite loop */
517   for(;;)
518   {
519       // Exercise 3
520       /*
521       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);//Enable RED LED
522       osDelayUntil(&TaskTimeStamp, DelayTimeMsec); //Delay 2 sec . Function creat
523       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);//Disable Red LED
524       osDelayUntil(&TaskTimeStamp, DelayTimeMsec);
525       //osDelay(1);*/
526 
527       // Exercise 4
528       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,GPIO_PIN_SET);//enable Red LED
529 
530       for(int i=0; i<=160;i++)
531       {
532           HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);//enable Orange LED
533           osDelay(25);
534       }
535       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,GPIO_PIN_RESET);//disable Red LED
536       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_RESET);//disable Orange LED
537       osDelay(6000);
538   }
539   /* USER CODE END StartFlashRedLedTask */
540 }
```