AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
ICHEP; Computer Engineering and Software Systems
Spring 2021

June 29th,2021 | **Course Code: CSE 345** | **Time: 2 Hours**

Real Time and Embedded System Design

The Exam Consists of **5 Questions in 4 Pages** | **Total Marks: 40 Marks**

Important Rules:	**تعليمات هامة**
• Having a (mobile -Smart Watch- earphones) inside the examination hall is forbidden and is considered as a cheating behavior. • It is forbidden to have any references, notes, books, or any other materials even if it is not related to the exam content with you in the examination hall. • This is an answer sheet. • Assume missing data if any – Read it all well, at first.	• حيازة (المحمول- الساعات الذكية - سماعة الأذن) داخل لجنة الامتحان يعتبر حالة غش تستوجب العقاب . • لايسمح بدخول أي كتب أو ملازم أو أوراق داخل اللجنة والمخالفة تعتبر حالة غش. • هذه ورقة إجابة أيضا - أقرأها أولا جيدا – أفترض الناقص إن وجد

Question 1: (8 Marks)

Assume the following snippet of code/application that already had all necessary declarations, inclusions, and prototypes. **In the given table, order the first 8 break points (PB) to be hit, when GO is pressed. At each PB, Define the states of all tasks.**

Break Point at Line?	State of Sender 1	State of Sender 2	State of Receiver

```
59  int main( void )
60  {
61    xQueue = xQueueCreate( 1, sizeof( long ) );
62    if( xQueue != NULL )
63    {
64      xTaskCreate( vSenderTask,"SENDER1", 240, ( void * ) 100, 3, NULL );
65      xTaskCreate( vSenderTask,"SENDER2", 240, ( void * ) 200, 2, NULL );
66      xTaskCreate( vReceiverTask, NULL, 240, NULL, 1, NULL );
67      vTaskStartScheduler();
68    }
69    for( ;; );
70  }
71  static void vSenderTask( void *pvParameters )
72  {
73  long lValueToSend;
74    lValueToSend = ( long ) pvParameters;
75    for( ;; )
76    {
77      xQueueSendToBack( xQueue, &lValueToSend, 100 / portTICK_RATE_MS );
78      taskYIELD();
79    }
80  }
81  static void vReceiverTask( void *pvParameters )
82  {
83  long lReceivedValue;
84    for( ;; )
85    {
86      xQueueReceive( xQueue, &lReceivedValue, 100 / portTICK_RATE_MS );
87      vPrintStringAndNumber( "Received = ", lReceivedValue );
88    }
89  }
```

Question 2: (10 Marks)

Assume the following snippet of code/application that already had all necessary declarations, inclusions, and prototypes. **In the given table, order the first 10 break points to be hit, when GO is pressed.**

1st Break Point Hit	2nd Break Point Hit	3rd Break Point Hit	4th Break Point Hit	5th Break Point Hit

6th Break Point Hit	7th Break Point Hit	8th Break Point Hit	9th Break Point Hit	10th Break Point Hit

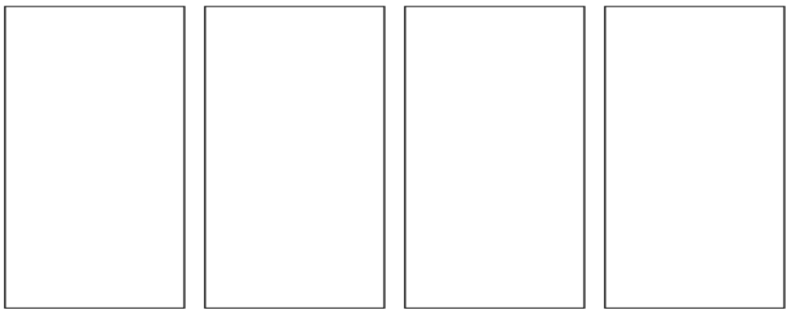
```
82  int main( void )
83  {
84      vSemaphoreCreateBinary( xBinarySemaphore );
85      if( xBinarySemaphore != NULL )
86      {
87          prvSetupSoftwareInterrupt();
88          xTaskCreate( vHandlerTask, NULL , 240, NULL,3, NULL );
89          xTaskCreate( vPeriodicTask, NULL, 240, NULL,1, NULL );
90          vTaskStartScheduler();
91      }
92  }
93  static void vHandlerTask( void *pvParameters )
94  {
95      xSemaphoreTake( xBinarySemaphore, 0 );
96      for( ;; )
97      {
98          xSemaphoreTake( xBinarySemaphore, portMAX_DELAY );
99          vPrintString( "Handler task - Processing event.\n" );
100     }
101 }
102 static void vPeriodicTask( void *pvParameters )
103 {
104     for( ;; )
105     {
106         vTaskDelay( 500 / portTICK_RATE_MS );
107         vPrintString( "Periodic task - About to generate an interrupt.\n" );
108         mainTRIGGER_INTERRUPT();
109         vPrintString( "Periodic task - Interrupt generated.\n\n" );
110     }
111 }
112 void vSoftwareInterruptHandler( void )
113 {
114 portBASE_TYPE xHigherPriorityTaskWoken = pdFALSE;
115     xSemaphoreGiveFromISR( xBinarySemaphore, &xHigherPriorityTaskWoken );
116     mainCLEAR_INTERRUPT();
117     portEND_SWITCHING_ISR( xHigherPriorityTaskWoken );
118 }
```

Question 3: (8 Marks)

Assume the following snippet of code/application that already had all necessary declarations, inclusions, and prototypes. Break points at lines 3, 12, 13 and 19 are shown**. Choose a suitable FreeRTOS Heap memory algorithm. Show in the figures below how Heap memory looks like at break points 12,13, and 14.** Start from a blank Heap at BP3.

```
1  int main( void )
2  {
3    xTaskCreate( vTask, "Task 1", 240, NULL, 2, NULL );
4    xTaskCreate( vTask, "Task 3", 240, NULL, 1, &xTask3Handle );
5    vTaskStartScheduler();
6    for( ;; );
7  }
8  void vTask( void *pvParameters )
9  {
10   for( ;; )
11   {
12     xTaskCreate( vTask2, "Task 2", 240, NULL, 3, NULL );
13     vTaskDelay( 100 / portTICK_RATE_MS );
14   }
15 }
16 void vTask2( void *pvParameters )
17 {
18   vTaskDelete( xTask3Handle );
19   vTaskDelay( 100 / portTICK_RATE_MS );
20 }
```

BP 3 | BP 12 | BP13 | BP19



Question 4: (6 Marks)

In a FreeRTOS project, three short tasks were created (Task A, Task B and Task C). Task A and Task B are having the periods 1Sec and 2Sec, respectively. Task C is a continuous task. Their priorities are 2, 3, and 1, respectively. **Sketch tasks timing diagram for the first 3 Seconds.** Vertical axe is the priority level while the horizontal axe is the time in seconds.

Question 5: (8 Marks)

Assume the following snippet of code/application that already had all necessary declarations, inclusions, and prototypes. Break points at lines 15, 26, 28, 42, 44, and 53 are shown**. In the given table, order the first five break points to be hit, when GO is pressed.**

1st Break Point Hit	2nd Break Point Hit	3rd Break Point Hit	4th Break Point Hit	5th Break Point Hit

```
1  int main( void ){
2      xMutex = xSemaphoreCreateMutex();
3      yMutex = xSemaphoreCreateMutex();
4      if( xMutex != NULL && yMutex != NULL )
5      {
6          xTaskCreate( prvPrintTask1, "Print1", 240, "Task 1 ************\n", 3, NULL );
7          xTaskCreate( prvPrintTask2, "Print2", 240, "Task 2 ------------\n", 2, &xTask2Handle );
8          xTaskCreate( vContinuousTask, "Print3", 240, NULL, 1, NULL );
9          vTaskStartScheduler();
10     }
11 }
12 static void prvNewPrintString( const portCHAR *pcString ){
13 static char cBuffer[ mainMAX_MSG_LEN ];
14     {
15         sprintf( cBuffer, "%s", pcString );
16         printf( cBuffer );
17     }
18 }
19 static void prvPrintTask1( void *pvParameters ){
20 char *pcStringToPrint;
21 unsigned portBASE_TYPE uxPriority;
22 uxPriority = uxTaskPriorityGet( NULL );
23 pcStringToPrint = ( char * ) pvParameters;
24     for( ;; )
25     {
26         xSemaphoreTake( xMutex, portMAX_DELAY );
27         vTaskPrioritySet(xTask2Handle,(uxPriority+1));
28         xSemaphoreTake (yMutex, portMAX_DELAY);
29         prvNewPrintString( pcStringToPrint );
30         xSemaphoreGive( xMutex );
31         xSemaphoreGive( yMutex );
32         vTaskDelay(  100 );
33     }
34 }
35 static void prvPrintTask2( void *pvParameters ){
36 char *pcStringToPrint;
37 unsigned portBASE_TYPE uxPriority;
38 uxPriority = uxTaskPriorityGet( NULL );
39 pcStringToPrint = ( char * ) pvParameters;
40     for( ;; )
41     {
42         xSemaphoreTake (yMutex, portMAX_DELAY);
43         xSemaphoreTake( xMutex, portMAX_DELAY );
44         prvNewPrintString( pcStringToPrint );
45         xSemaphoreGive( yMutex );
46         xSemaphoreGive( xMutex );
47         vTaskPrioritySet(NULL,(uxPriority-2));
48     }
49 }
50 static void vContinuousTask( void *pvParameters ){
51     for( ;; )
52     {
53         vPrintString( " task3 is running.....\n" );
54         vTaskDelay( 100 );
55     }
56 }
```