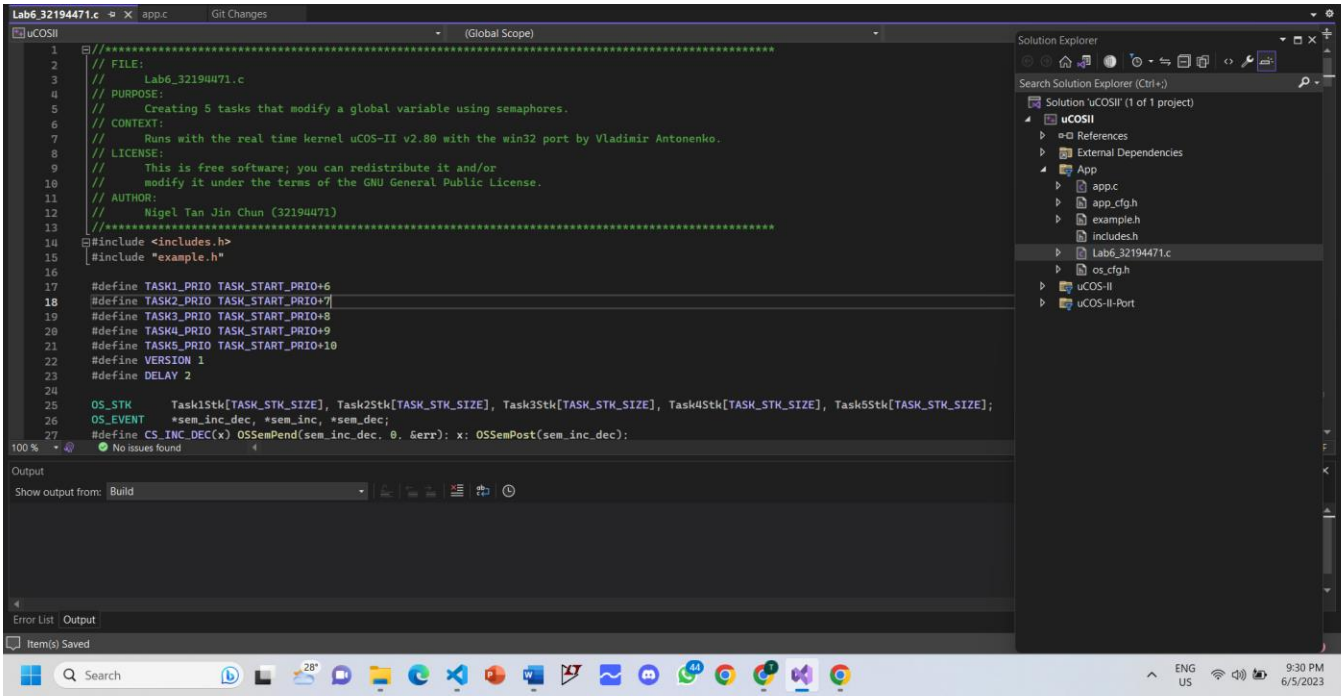
Assessment (ECE3073)

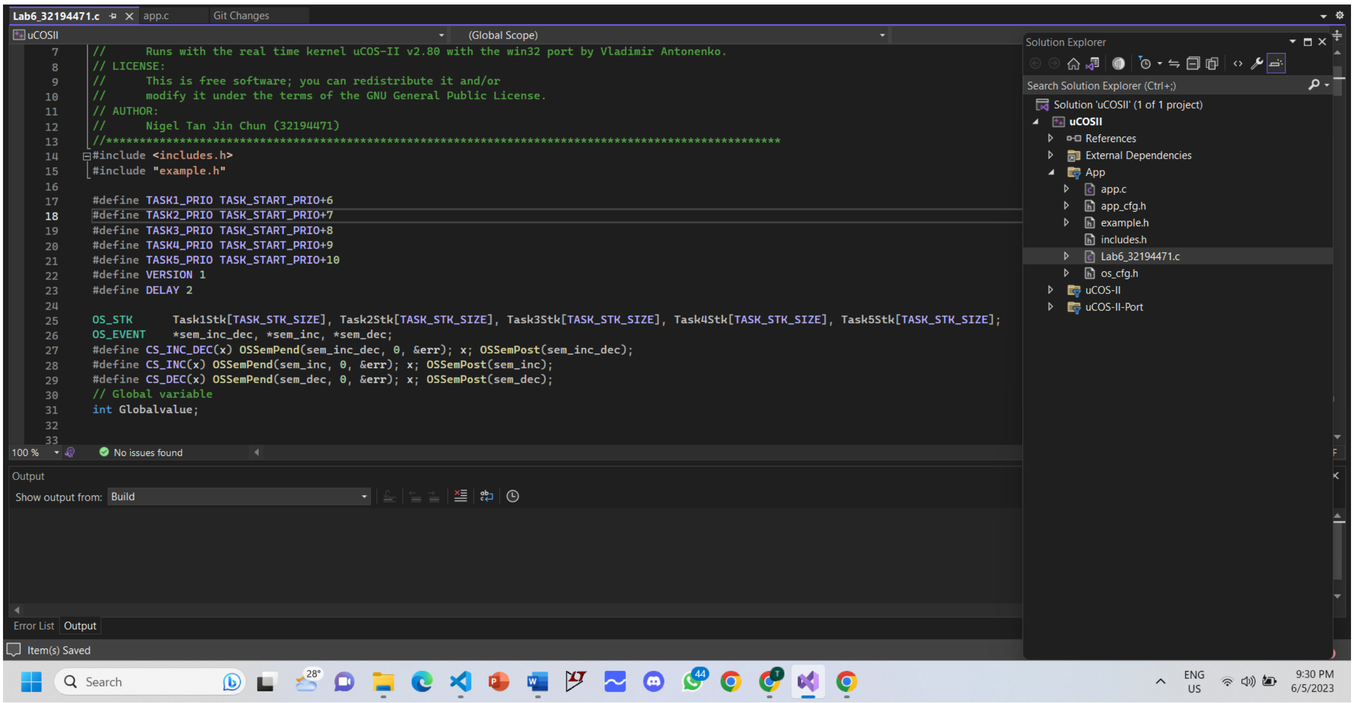
Lab 6 Code

Name: Tan Jin Chun
Student ID: 32194471

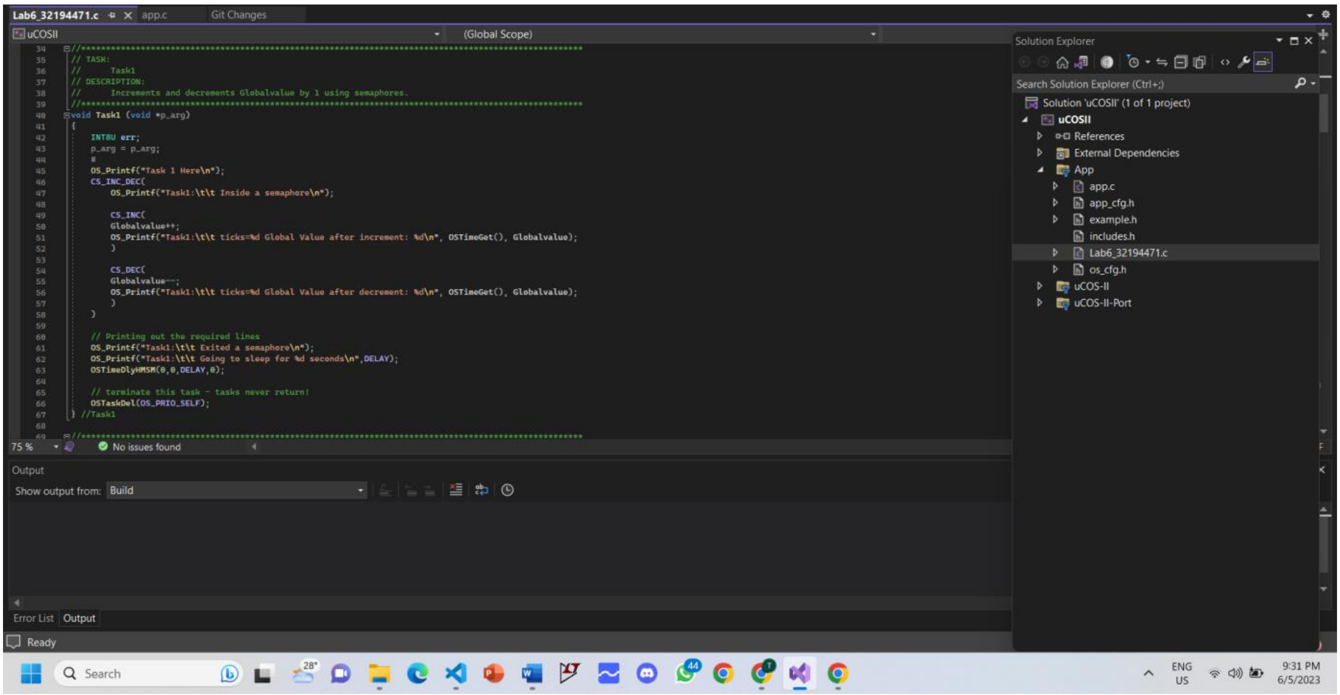
Screenshot of my Code



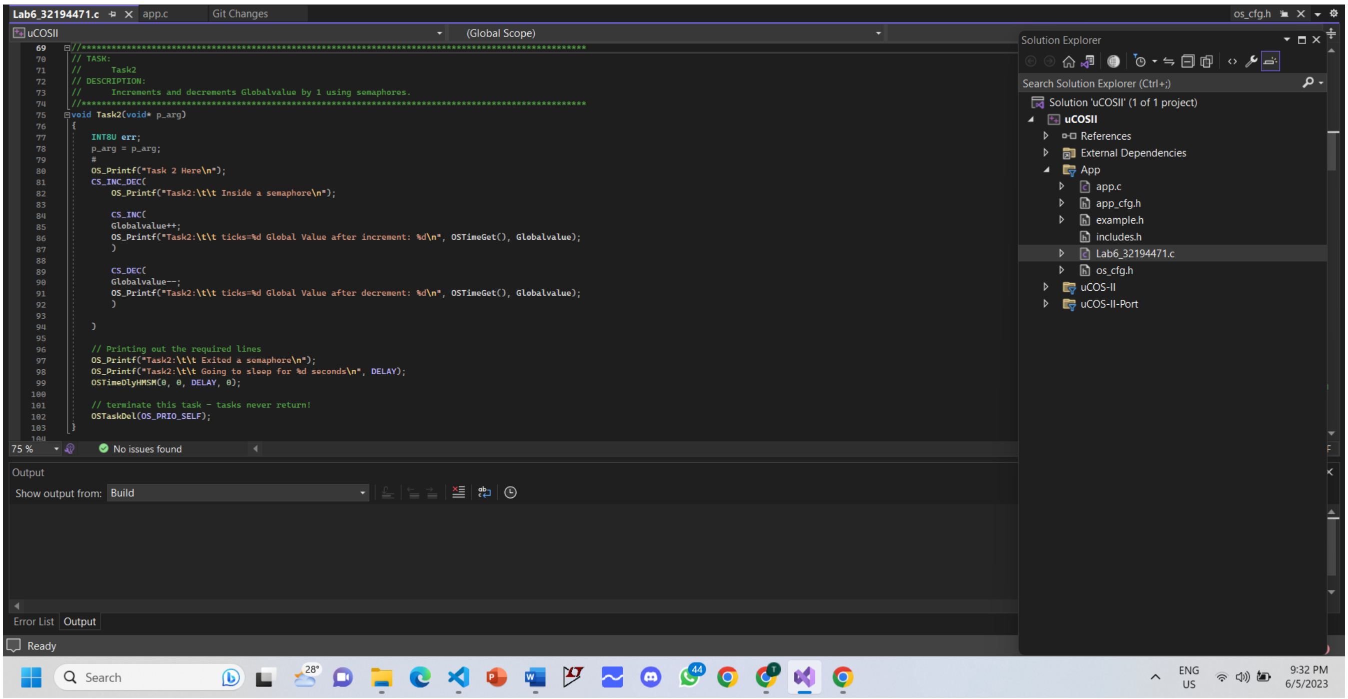
```c
//**********************************************************************************************************
// FILE:
//      Lab6_32194471.c
// PURPOSE:
//      Creating 5 tasks that modify a global variable using semaphores.
// CONTEXT:
//      Runs with the real time kernel uCOS-II v2.80 with the win32 port by Vladimir Antonenko.
// LICENSE:
//      This is free software; you can redistribute it and/or
//      modify it under the terms of the GNU General Public License.
// AUTHOR:
//      Nigel Tan Jin Chun (32194471)
//**********************************************************************************************************
#include <includes.h>
#include "example.h"

#define TASK1_PRIO TASK_START_PRIO+6
#define TASK2_PRIO TASK_START_PRIO+7
#define TASK3_PRIO TASK_START_PRIO+8
#define TASK4_PRIO TASK_START_PRIO+9
#define TASK5_PRIO TASK_START_PRIO+10
#define VERSION 1
#define DELAY 2

OS_STK      Task1Stk[TASK_STK_SIZE], Task2Stk[TASK_STK_SIZE], Task3Stk[TASK_STK_SIZE], Task4Stk[TASK_STK_SIZE], Task5Stk[TASK_STK_SIZE];
OS_EVENT    *sem_inc_dec, *sem_inc, *sem_dec;
#define CS_INC_DEC(x) OSSemPend(sem_inc_dec, 0, &err); x; OSSemPost(sem_inc_dec);
```



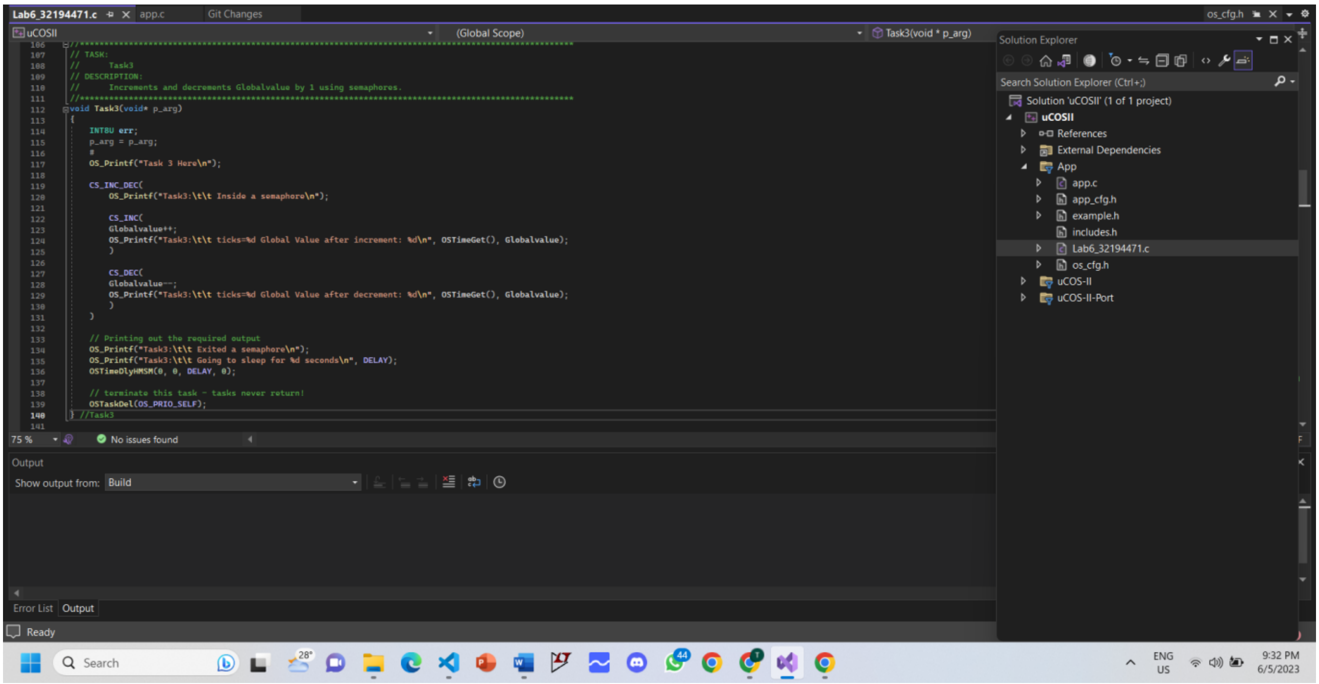
```c
//      Runs with the real time kernel uCOS-II v2.80 with the win32 port by Vladimir Antonenko.
// LICENSE:
//      This is free software; you can redistribute it and/or
//      modify it under the terms of the GNU General Public License.
// AUTHOR:
//      Nigel Tan Jin Chun (32194471)
//**********************************************************************************************************
#include <includes.h>
#include "example.h"

#define TASK1_PRIO TASK_START_PRIO+6
#define TASK2_PRIO TASK_START_PRIO+7
#define TASK3_PRIO TASK_START_PRIO+8
#define TASK4_PRIO TASK_START_PRIO+9
#define TASK5_PRIO TASK_START_PRIO+10
#define VERSION 1
#define DELAY 2

OS_STK      Task1Stk[TASK_STK_SIZE], Task2Stk[TASK_STK_SIZE], Task3Stk[TASK_STK_SIZE], Task4Stk[TASK_STK_SIZE], Task5Stk[TASK_STK_SIZE];
OS_EVENT    *sem_inc_dec, *sem_inc, *sem_dec;
#define CS_INC_DEC(x) OSSemPend(sem_inc_dec, 0, &err); x; OSSemPost(sem_inc_dec);
#define CS_INC(x) OSSemPend(sem_inc, 0, &err); x; OSSemPost(sem_inc);
#define CS_DEC(x) OSSemPend(sem_dec, 0, &err); x; OSSemPost(sem_dec);
// Global variable
int Globalvalue;
```



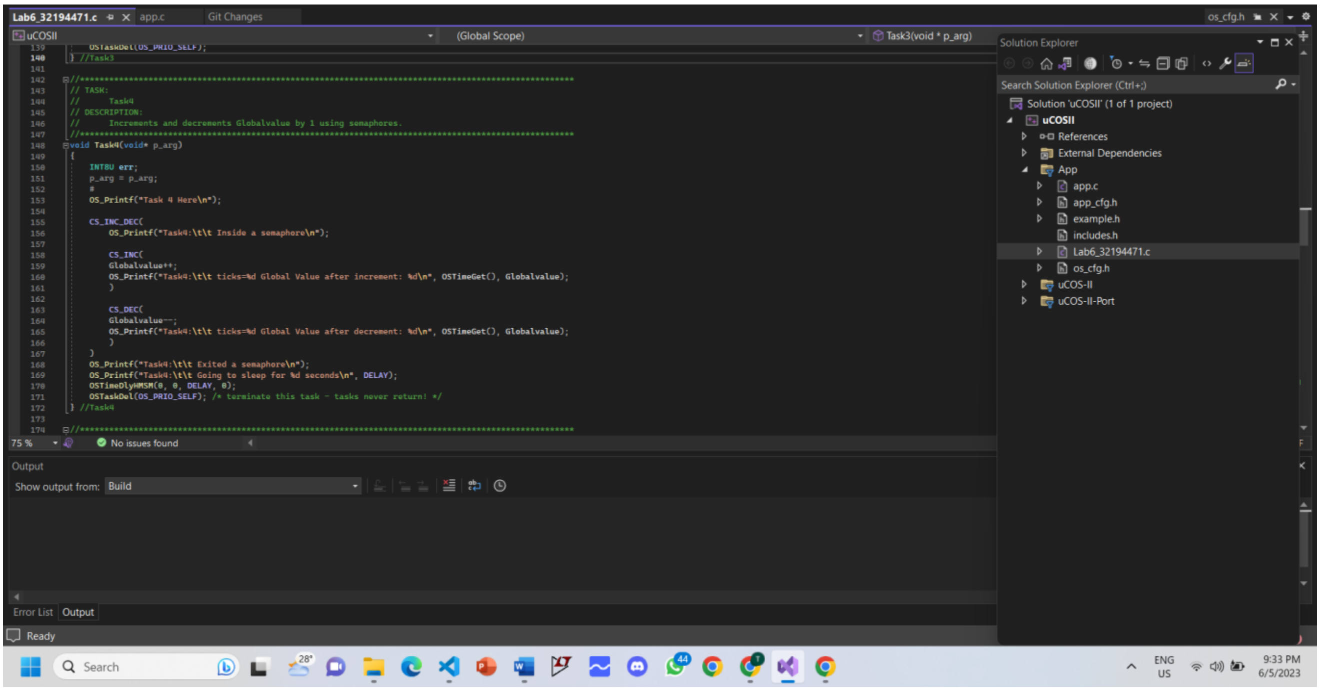
```c
//**************************************************************************************************
// TASK:
//      Task1
// DESCRIPTION:
//      Increments and decrements Globalvalue by 1 using semaphores.
//**************************************************************************************************
void Task1 (void *p_arg)
{
    INT8U err;
    p_arg = p_arg;
    #
    OS_Printf("Task 1 Here\n");
    CS_INC_DEC(
        OS_Printf("Task1:\t\t Inside a semaphore\n");

        CS_INC(
        Globalvalue++;
        OS_Printf("Task1:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
        )

        CS_DEC(
        Globalvalue--;
        OS_Printf("Task1:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )
    )

    // Printing out the required lines
    OS_Printf("Task1:\t\t Exited a semaphore\n");
    OS_Printf("Task1:\t\t Going to sleep for %d seconds\n",DELAY);
    OSTimeDlyHMSM(0,0,DELAY,0);

    // terminate this task - tasks never return!
    OSTaskDel(OS_PRIO_SELF);
} //Task1
```



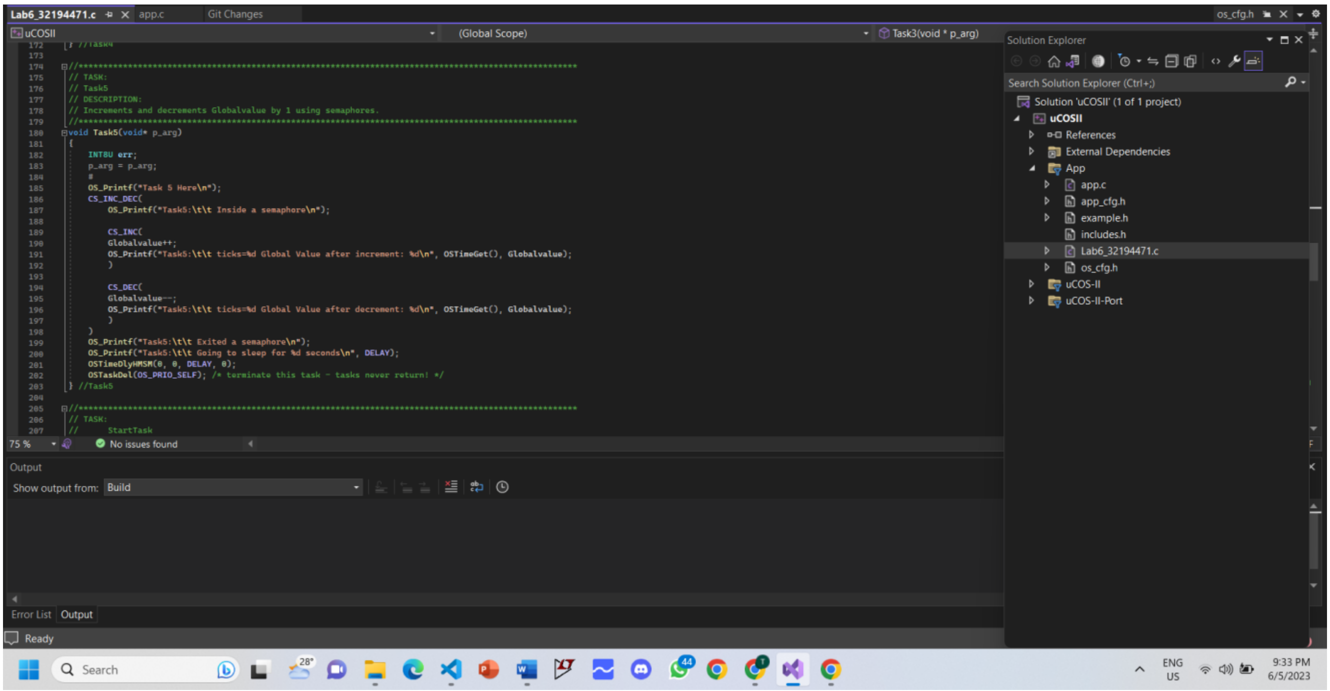
```c
//**************************************************************************************************
// TASK:
//      Task2
// DESCRIPTION:
//      Increments and decrements Globalvalue by 1 using semaphores.
//**************************************************************************************************
void Task2(void* p_arg)
{
    INT8U err;
    p_arg = p_arg;
    #
    OS_Printf("Task 2 Here\n");
    CS_INC_DEC(
        OS_Printf("Task2:\t\t Inside a semaphore\n");

        CS_INC(
        Globalvalue++;
        OS_Printf("Task2:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
        )

        CS_DEC(
        Globalvalue--;
        OS_Printf("Task2:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )

    )

    // Printing out the required lines
    OS_Printf("Task2:\t\t Exited a semaphore\n");
    OS_Printf("Task2:\t\t Going to sleep for %d seconds\n", DELAY);
    OSTimeDlyHMSM(0, 0, DELAY, 0);

    // terminate this task - tasks never return!
    OSTaskDel(OS_PRIO_SELF);
}
```



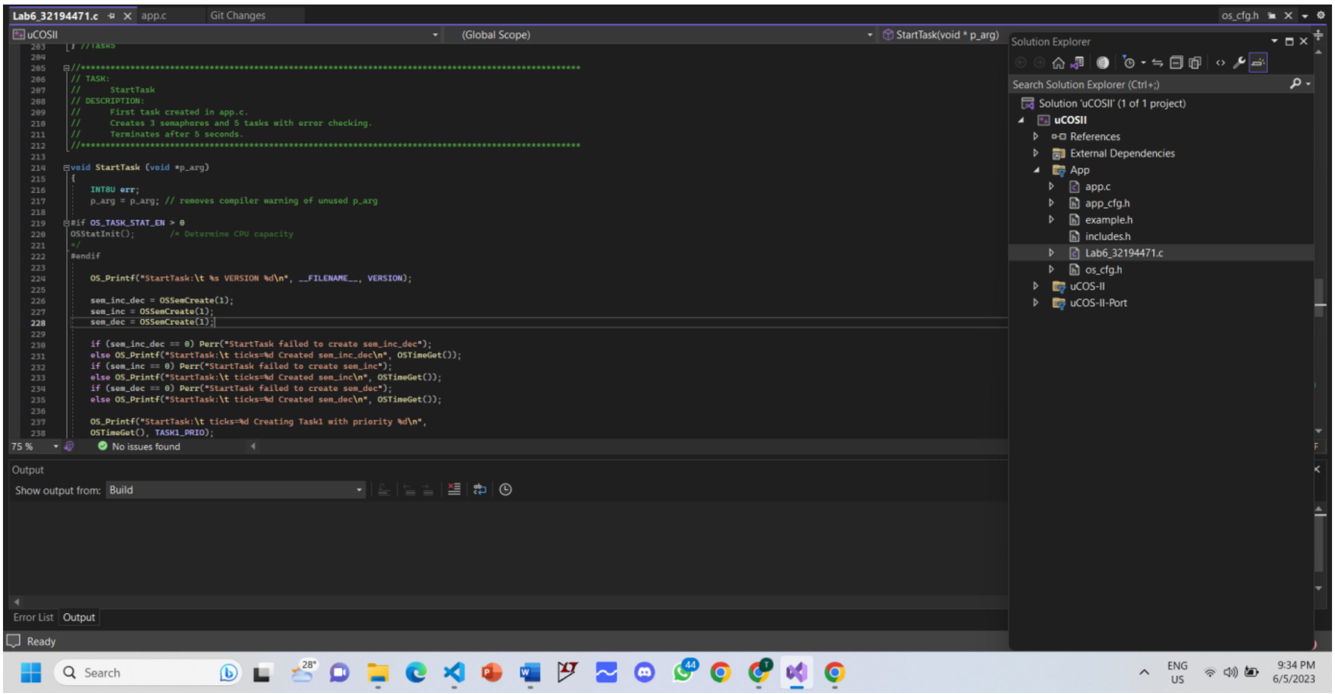
```c
//*******************************************************************************************
// TASK:
//      Task3
// DESCRIPTION:
//      Increments and decrements Globalvalue by 1 using semaphores.
//*******************************************************************************************
void Task3(void* p_arg)
{
    INT8U err;
    p_arg = p_arg;
    #
    OS_Printf("Task 3 Here\n");

    CS_INC_DEC(
        OS_Printf("Task3:\t\t Inside a semaphore\n");

        CS_INC(
        Globalvalue++;
        OS_Printf("Task3:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
        )

        CS_DEC(
        Globalvalue--;
        OS_Printf("Task3:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )
    )

    // Printing out the required output
    OS_Printf("Task3:\t\t Exited a semaphore\n");
    OS_Printf("Task3:\t\t Going to sleep for %d seconds\n", DELAY);
    OSTimeDlyHMSM(0, 0, DELAY, 0);

    // terminate this task - tasks never return!
    OSTaskDel(OS_PRIO_SELF);
} //Task3
```



```c
    OSTaskDel(OS_PRIO_SELF);
} //Task3

//*******************************************************************************************
// TASK:
//      Task4
// DESCRIPTION:
//      Increments and decrements Globalvalue by 1 using semaphores.
//*******************************************************************************************
void Task4(void* p_arg)
{
    INT8U err;
    p_arg = p_arg;
    #
    OS_Printf("Task 4 Here\n");

    CS_INC_DEC(
        OS_Printf("Task4:\t\t Inside a semaphore\n");

        CS_INC(
        Globalvalue++;
        OS_Printf("Task4:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
        )

        CS_DEC(
        Globalvalue--;
        OS_Printf("Task4:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )
    )
    OS_Printf("Task4:\t\t Exited a semaphore\n");
    OS_Printf("Task4:\t\t Going to sleep for %d seconds\n", DELAY);
    OSTimeDlyHMSM(0, 0, DELAY, 0);
    OSTaskDel(OS_PRIO_SELF); /* terminate this task - tasks never return! */
} //Task4

//*******************************************************************************************
```

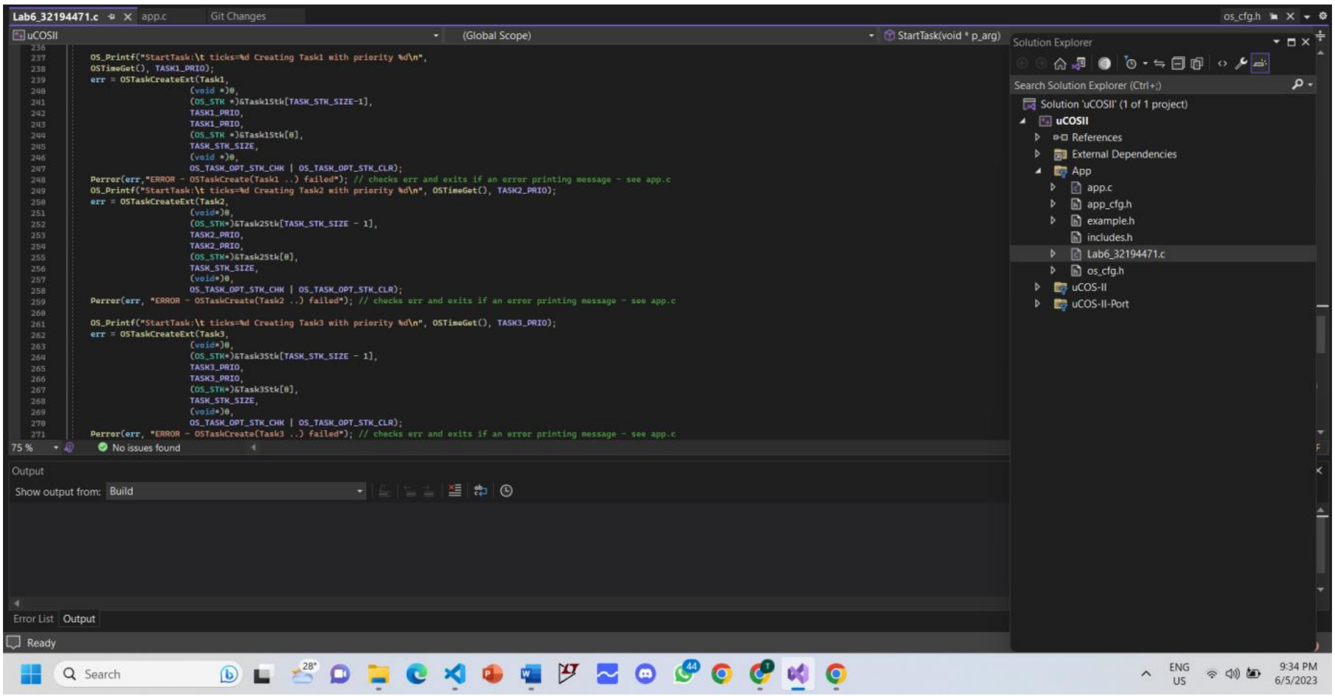


```
} //Task4

//*****************************************************************************************************
// TASK:
// Task5
// DESCRIPTION:
// Increments and decrements Globalvalue by 1 using semaphores.
//*****************************************************************************************************
void Task5(void* p_arg)
{
    INT8U err;
    p_arg = p_arg;
    #
    OS_Printf("Task 5 Here\n");
    CS_INC_DEC(
        OS_Printf("Task5:\t\t Inside a semaphore\n");

        CS_INC(
        Globalvalue++;
        OS_Printf("Task5:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
        )

        CS_DEC(
        Globalvalue--;
        OS_Printf("Task5:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )
    )
    OS_Printf("Task5:\t\t Exited a semaphore\n");
    OS_Printf("Task5:\t\t Going to sleep for %d seconds\n", DELAY);
    OSTimeDlyHMSM(0, 0, DELAY, 0);
    OSTaskDel(OS_PRIO_SELF); /* terminate this task - tasks never return! */
} //Task5

//*****************************************************************************************************
// TASK:
//      StartTask
```

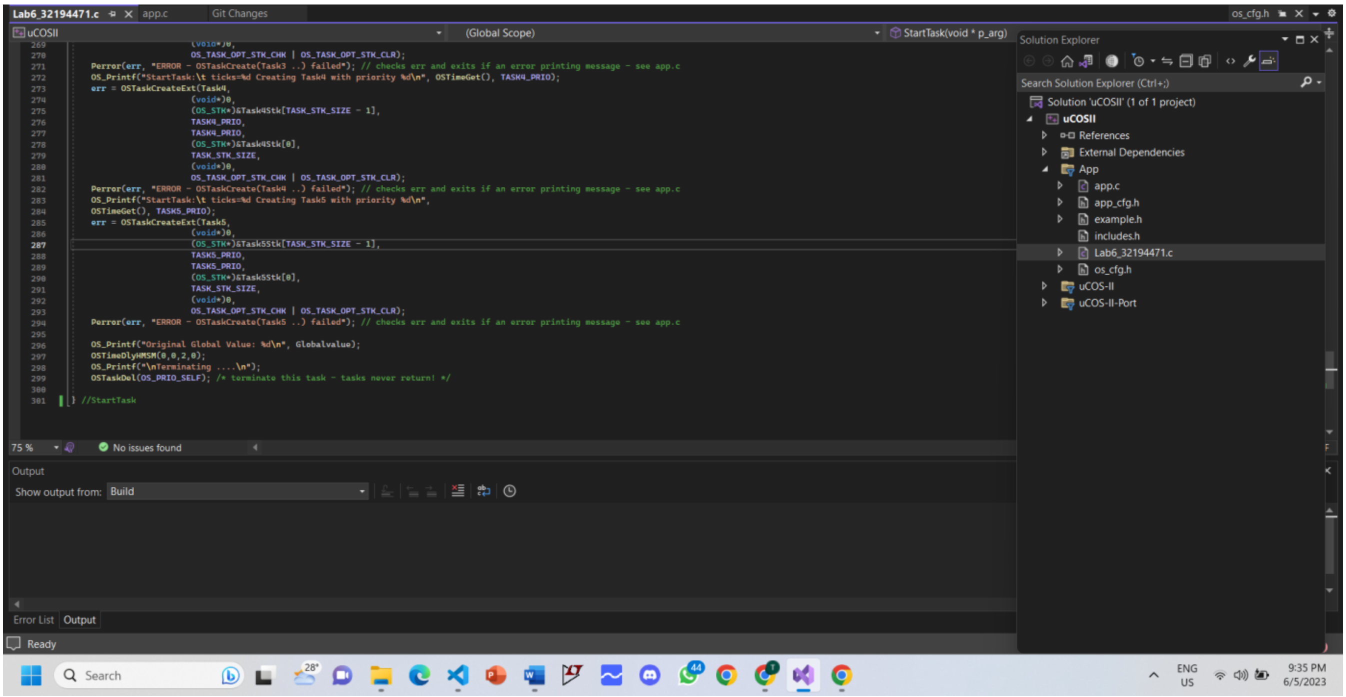


```
} //Task5

//*****************************************************************************************************
// TASK:
//      StartTask
// DESCRIPTION:
//      First task created in app.c.
//      Creates 3 semaphores and 5 tasks with error checking.
//      Terminates after 5 seconds.
//*****************************************************************************************************

void StartTask (void *p_arg)
{
    INT8U err;
    p_arg = p_arg; // removes compiler warning of unused p_arg

#if OS_TASK_STAT_EN > 0
OSStatInit();       /* Determine CPU capacity
*/
#endif

    OS_Printf("StartTask:\t %s VERSION %d\n", __FILENAME__, VERSION);

    sem_inc_dec = OSSemCreate(1);
    sem_inc = OSSemCreate(1);
    sem_dec = OSSemCreate(1);

    if (sem_inc_dec == 0) Perr("StartTask failed to create sem_inc_dec");
    else OS_Printf("StartTask:\t ticks=%d Created sem_inc_dec\n", OSTimeGet());
    if (sem_inc == 0) Perr("StartTask failed to create sem_inc");
    else OS_Printf("StartTask:\t ticks=%d Created sem_inc\n", OSTimeGet());
    if (sem_dec == 0) Perr("StartTask failed to create sem_dec");
    else OS_Printf("StartTask:\t ticks=%d Created sem_dec\n", OSTimeGet());

    OS_Printf("StartTask:\t ticks=%d Creating Task1 with priority %d\n",
    OSTimeGet(), TASK1_PRIO);
```

`



```
    OS_Printf("StartTask:\t ticks=%d Creating Task1 with priority %d\n",
    OSTimeGet(), TASK1_PRIO);
    err = OSTaskCreateExt(Task1,
                    (void *)0,
                    (OS_STK *)&Task1Stk[TASK_STK_SIZE-1],
                    TASK1_PRIO,
                    TASK1_PRIO,
                    (OS_STK *)&Task1Stk[0],
                    TASK_STK_SIZE,
                    (void *)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
    Perror(err,"ERROR - OSTaskCreate(Task1 ..) failed"); // checks err and exits if an error printing message - see app.c
    OS_Printf("StartTask:\t ticks=%d Creating Task2 with priority %d\n", OSTimeGet(), TASK2_PRIO);
    err = OSTaskCreateExt(Task2,
                    (void*)0,
                    (OS_STK*)&Task2Stk[TASK_STK_SIZE - 1],
                    TASK2_PRIO,
                    TASK2_PRIO,
                    (OS_STK*)&Task2Stk[0],
                    TASK_STK_SIZE,
                    (void*)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
    Perror(err, "ERROR - OSTaskCreate(Task2 ..) failed"); // checks err and exits if an error printing message - see app.c

    OS_Printf("StartTask:\t ticks=%d Creating Task3 with priority %d\n", OSTimeGet(), TASK3_PRIO);
    err = OSTaskCreateExt(Task3,
                    (void*)0,
                    (OS_STK*)&Task3Stk[TASK_STK_SIZE - 1],
                    TASK3_PRIO,
                    TASK3_PRIO,
                    (OS_STK*)&Task3Stk[0],
                    TASK_STK_SIZE,
                    (void*)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
    Perror(err, "ERROR - OSTaskCreate(Task3 ..) failed"); // checks err and exits if an error printing message - see app.c
```



```
                    (void*)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
    Perror(err, "ERROR - OSTaskCreate(Task3 ..) failed"); // checks err and exits if an error printing message - see app.c
    OS_Printf("StartTask:\t ticks=%d Creating Task4 with priority %d\n", OSTimeGet(), TASK4_PRIO);
    err = OSTaskCreateExt(Task4,
                    (void*)0,
                    (OS_STK*)&Task4Stk[TASK_STK_SIZE - 1],
                    TASK4_PRIO,
                    TASK4_PRIO,
                    (OS_STK*)&Task4Stk[0],
                    TASK_STK_SIZE,
                    (void*)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
    Perror(err, "ERROR - OSTaskCreate(Task4 ..) failed"); // checks err and exits if an error printing message - see app.c
    OS_Printf("StartTask:\t ticks=%d Creating Task5 with priority %d\n",
    OSTimeGet(), TASK5_PRIO);
    err = OSTaskCreateExt(Task5,
                    (void*)0,
                    (OS_STK*)&Task5Stk[TASK_STK_SIZE - 1],
                    TASK5_PRIO,
                    TASK5_PRIO,
                    (OS_STK*)&Task5Stk[0],
                    TASK_STK_SIZE,
                    (void*)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
    Perror(err, "ERROR - OSTaskCreate(Task5 ..) failed"); // checks err and exits if an error printing message - see app.c

    OS_Printf("Original Global Value: %d\n", Globalvalue);
    OSTimeDlyHMSM(0,0,2,0);
    OS_Printf("\nTerminating ....\n");
    OSTaskDel(OS_PRIO_SELF); /* terminate this task - tasks never return! */

} //StartTask
```

Actual Code

```
//**********************************************************************************
*********************

// FILE:

//     Lab6_32194471.c

// PURPOSE:

//     Creating 5 tasks that modify a global variable using semaphores.

// CONTEXT:

//     Runs with the real time kernel uCOS-II v2.80 with the win32 port by Vladimir Antonenko.

// LICENSE:

//     This is free software; you can redistribute it and/or

//     modify it under the terms of the GNU General Public License.

// AUTHOR:

//     Nigel Tan Jin Chun (32194471)

//**********************************************************************************
*********************

#include <includes.h>

#include "example.h"


#define TASK1_PRIO TASK_START_PRIO+6

#define TASK2_PRIO TASK_START_PRIO+7

#define TASK3_PRIO TASK_START_PRIO+8

#define TASK4_PRIO TASK_START_PRIO+9

#define TASK5_PRIO TASK_START_PRIO+10

#define VERSION 1

#define DELAY 2


OS_STK     Task1Stk[TASK_STK_SIZE], Task2Stk[TASK_STK_SIZE], Task3Stk[TASK_STK_SIZE],
Task4Stk[TASK_STK_SIZE], Task5Stk[TASK_STK_SIZE];

OS_EVENT    *sem_inc_dec, *sem_inc, *sem_dec;

#define CS_INC_DEC(x) OSSemPend(sem_inc_dec, 0, &err); x; OSSemPost(sem_inc_dec);

#define CS_INC(x) OSSemPend(sem_inc, 0, &err); x; OSSemPost(sem_inc);
```

```
#define CS_DEC(x) OSSemPend(sem_dec, 0, &err); x; OSSemPost(sem_dec);
// Global variable
int Globalvalue;



//********************************************************************************
*********************
// TASK:
//     Task1
// DESCRIPTION:
//     Increments and decrements Globalvalue by 1 using semaphores.
//********************************************************************************
*********************
void Task1 (void *p_arg)
{
  INT8U err;
  p_arg = p_arg;
  #
  OS_Printf("Task 1 Here\n");
  CS_INC_DEC(
    OS_Printf("Task1:\t\t Inside a semaphore\n");

    CS_INC(
    Globalvalue++;
    OS_Printf("Task1:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
    )

    CS_DEC(
    Globalvalue--;
    OS_Printf("Task1:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
    )
  )
```

```
   // Printing out the required lines
   OS_Printf("Task1:\t\t Exited a semaphore\n");
   OS_Printf("Task1:\t\t Going to sleep for %d seconds\n",DELAY);
   OSTimeDlyHMSM(0,0,DELAY,0);

   // terminate this task - tasks never return!
   OSTaskDel(OS_PRIO_SELF);
} //Task1

//****************************************************************************
********************
// TASK:
//     Task2
// DESCRIPTION:
//     Increments and decrements Globalvalue by 1 using semaphores.
//****************************************************************************
********************
void Task2(void* p_arg)
{
   INT8U err;
   p_arg = p_arg;
   #
   OS_Printf("Task 2 Here\n");
   CS_INC_DEC(
      OS_Printf("Task2:\t\t Inside a semaphore\n");

      CS_INC(
      Globalvalue++;
      OS_Printf("Task2:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
      )
```

```
        CS_DEC(
        Globalvalue--;
        OS_Printf("Task2:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )

    )

    // Printing out the required lines
    OS_Printf("Task2:\t\t Exited a semaphore\n");
    OS_Printf("Task2:\t\t Going to sleep for %d seconds\n", DELAY);
    OSTimeDlyHMSM(0, 0, DELAY, 0);

    // terminate this task - tasks never return!
    OSTaskDel(OS_PRIO_SELF);
}



//*****************************************************************************
********************
// TASK:
//     Task3
// DESCRIPTION:
//     Increments and decrements Globalvalue by 1 using semaphores.
//*****************************************************************************
********************
void Task3(void* p_arg)
{
    INT8U err;
    p_arg = p_arg;
    #
    OS_Printf("Task 3 Here\n");
    CS_INC_DEC(
        OS_Printf("Task3:\t\t Inside a semaphore\n");

        CS_INC(
        Globalvalue++;
        OS_Printf("Task3:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
        )

        CS_DEC(
        Globalvalue--;
        OS_Printf("Task3:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
        )
    )

    // Printing out the required output
    OS_Printf("Task3:\t\t Exited a semaphore\n");
    OS_Printf("Task3:\t\t Going to sleep for %d seconds\n", DELAY);
    OSTimeDlyHMSM(0, 0, DELAY, 0);

    // terminate this task - tasks never return!
    OSTaskDel(OS_PRIO_SELF);
} //Task3

//**************************************************************************************
*********************
// TASK:
//      Task4
// DESCRIPTION:
//      Increments and decrements Globalvalue by 1 using semaphores.
//**************************************************************************************
*********************
void Task4(void* p_arg)
```

```
{
   INT8U err;
   p_arg = p_arg;
   #
   OS_Printf("Task 4 Here\n");

   CS_INC_DEC(
      OS_Printf("Task4:\t\t Inside a semaphore\n");

      CS_INC(
      Globalvalue++;
      OS_Printf("Task4:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
      )

      CS_DEC(
      Globalvalue--;
      OS_Printf("Task4:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
      )
   )
   OS_Printf("Task4:\t\t Exited a semaphore\n");
   OS_Printf("Task4:\t\t Going to sleep for %d seconds\n", DELAY);
   OSTimeDlyHMSM(0, 0, DELAY, 0);
   OSTaskDel(OS_PRIO_SELF); /* terminate this task - tasks never return! */
} //Task4

//********************************************************************************
********************
// TASK:
// Task5
// DESCRIPTION:
// Increments and decrements Globalvalue by 1 using semaphores.
```

```
//******************************************************************************
********************
void Task5(void* p_arg)
{
  INT8U err;
  p_arg = p_arg;
  #
  OS_Printf("Task 5 Here\n");
  CS_INC_DEC(
    OS_Printf("Task5:\t\t Inside a semaphore\n");

    CS_INC(
    Globalvalue++;
    OS_Printf("Task5:\t\t ticks=%d Global Value after increment: %d\n", OSTimeGet(), Globalvalue);
    )

    CS_DEC(
    Globalvalue--;
    OS_Printf("Task5:\t\t ticks=%d Global Value after decrement: %d\n", OSTimeGet(), Globalvalue);
    )
  )
  OS_Printf("Task5:\t\t Exited a semaphore\n");
  OS_Printf("Task5:\t\t Going to sleep for %d seconds\n", DELAY);
  OSTimeDlyHMSM(0, 0, DELAY, 0);
  OSTaskDel(OS_PRIO_SELF); /* terminate this task - tasks never return! */
} //Task5

//******************************************************************************
********************
// TASK:
//     StartTask
// DESCRIPTION:
```

```
//      First task created in app.c.
//      Creates 3 semaphores and 5 tasks with error checking.
//      Terminates after 5 seconds.
//*****************************************************************************
********************

void StartTask (void *p_arg)
{
   INT8U err;
   p_arg = p_arg; // removes compiler warning of unused p_arg

#if OS_TASK_STAT_EN > 0
OSStatInit();       /* Determine CPU capacity
*/
#endif

   OS_Printf("StartTask:\t %s VERSION %d\n", __FILENAME__, VERSION);

   sem_inc_dec = OSSemCreate(1);
   sem_inc = OSSemCreate(1);
   sem_dec = OSSemCreate(1);

   if (sem_inc_dec == 0) Perr("StartTask failed to create sem_inc_dec");
   else OS_Printf("StartTask:\t ticks=%d Created sem_inc_dec\n", OSTimeGet());
   if (sem_inc == 0) Perr("StartTask failed to create sem_inc");
   else OS_Printf("StartTask:\t ticks=%d Created sem_inc\n", OSTimeGet());
   if (sem_dec == 0) Perr("StartTask failed to create sem_dec");
   else OS_Printf("StartTask:\t ticks=%d Created sem_dec\n", OSTimeGet());

   OS_Printf("StartTask:\t ticks=%d Creating Task1 with priority %d\n",
   OSTimeGet(), TASK1_PRIO);
   err = OSTaskCreateExt(Task1,
                  (void *)0,
                  (OS_STK *)&Task1Stk[TASK_STK_SIZE-1],
                  TASK1_PRIO,
                  TASK1_PRIO,
                  (OS_STK *)&Task1Stk[0],
                  TASK_STK_SIZE,
                  (void *)0,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
   Perror(err,"ERROR - OSTaskCreate(Task1 ..) failed"); // checks err and exits if an error printing
message - see app.c
   OS_Printf("StartTask:\t ticks=%d Creating Task2 with priority %d\n", OSTimeGet(), TASK2_PRIO);
   err = OSTaskCreateExt(Task2,
                  (void*)0,
                  (OS_STK*)&Task2Stk[TASK_STK_SIZE - 1],
                  TASK2_PRIO,
                  TASK2_PRIO,
                  (OS_STK*)&Task2Stk[0],
                  TASK_STK_SIZE,
                  (void*)0,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
   Perror(err, "ERROR - OSTaskCreate(Task2 ..) failed"); // checks err and exits if an error printing
message - see app.c

   OS_Printf("StartTask:\t ticks=%d Creating Task3 with priority %d\n", OSTimeGet(), TASK3_PRIO);
   err = OSTaskCreateExt(Task3,
                  (void*)0,
                  (OS_STK*)&Task3Stk[TASK_STK_SIZE - 1],
                  TASK3_PRIO,
                  TASK3_PRIO,
                  (OS_STK*)&Task3Stk[0],
                  TASK_STK_SIZE,
                  (void*)0,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
   Perror(err, "ERROR - OSTaskCreate(Task3 ..) failed"); // checks err and exits if an error printing message - see app.c
   OS_Printf("StartTask:\t ticks=%d Creating Task4 with priority %d\n", OSTimeGet(), TASK4_PRIO);
   err = OSTaskCreateExt(Task4,
                  (void*)0,
                  (OS_STK*)&Task4Stk[TASK_STK_SIZE - 1],
                  TASK4_PRIO,
                  TASK4_PRIO,
                  (OS_STK*)&Task4Stk[0],
                  TASK_STK_SIZE,
                  (void*)0,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
   Perror(err, "ERROR - OSTaskCreate(Task4 ..) failed"); // checks err and exits if an error printing message - see app.c
   OS_Printf("StartTask:\t ticks=%d Creating Task5 with priority %d\n",
   OSTimeGet(), TASK5_PRIO);
   err = OSTaskCreateExt(Task5,
                  (void*)0,
                  (OS_STK*)&Task5Stk[TASK_STK_SIZE - 1],
                  TASK5_PRIO,
                  TASK5_PRIO,
                  (OS_STK*)&Task5Stk[0],
                  TASK_STK_SIZE,
                  (void*)0,
                  OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);
   Perror(err, "ERROR - OSTaskCreate(Task5 ..) failed"); // checks err and exits if an error printing message - see app.c

   OS_Printf("Original Global Value: %d\n", Globalvalue);
   OSTimeDlyHMSM(0,0,2,0);
```

```
    OS_Printf("\nTerminating ....\n");

    OSTaskDel(OS_PRIO_SELF); /* terminate this task - tasks never return! */


} //StartTask
```