WISEflow

Ioan Sorin Balcoianu

Participation

IT-ESW1

Exam

Thursday

JUN. 16, 2

09:00

Grade

My paper

© Copyright UNIwise 2012-2022

Support

My paper - Automatic validation

Below you can see the automatic validation of your answers

Item 1

Static 5 points

What will be written in the console if the following code is executed:
Right answer 5 points.

```
#include <stdio.h>
static y;

void foo(int x) {
    static z=10;
    y = x+2;
    z = z+y;
    printf("x is %d, y is %d, z is %d.\n", x, y, z);
}

int main() {
    foo(42);
    foo(-5);
    return 0;
}
```

A	x is: 42, y is 52, z is 62. x is: -5, y is 57, z is 119.	✓
B	x is: 42, y is 52, z is 62. x is: -5, y is 57, z is 67.	
C	x is: 42, y is 52, z is 62. x is: -5, y is 5, z is 15.	
D	x is: 42, y is 52, z is 62. x is: -5, y is 5, z is 72.	
E	Nothing, the code cannot compile	✗

Item 2

Pointers 2 (3 Points)

What will this code snipped print out on the console?

```
void foo()
{
    char myString[] = "The quick brown fox...";
    char *p = myString + 6;
    printf("1: %c, 2: %c, 3: %c", *p++, *p, *(myString+2));
}
```

One letter in each box.
All answers must be correct in order to score points.

1: 1 t ✓ 2: 2 c ✓ 3: 3 e ✓

Item 3

Strings (2 points)

Fill in the missing parts of this program so that it will print out *Chevrolet Corvette*

All answers must be correct in order to score points.

```
#include <stdio.h>

#include <string.h>

void strings()

{

  char *brand = "Chevrolet " ;

  char model[15];

  1 strcpy(model, "Corvette");

  char* myCar = calloc(1, 30);


  if (NULL != myCar)

  {

    2 strcpy(myCar, brand);

    3 strcat(myCar, model);

    printf("My car is a %s\n", myCar);

    4 free(myCar);

  }

}
```

Item 4

Insert the right code snippets so that taskC waits for taskA and taskB to both set their bits.

5 points if all three options are right otherwise 0.

```
// includes and defines ...
EventGroupHandle_t _myEventGroup = NULL;
#define BIT_TASK_A_READY (1 << 0)
#define BIT_TASK_B_READY (1 << 1)

// ------------------------------------------------------------------
void task_A(void* pvParameters)
{
    (void)pvParameters;
    for (;;)
    {
        // do something

1   xEventGroupSetBits(_myEventGroup, BIT_TASK_A_READY);

    }
}
// ------------------------------------------------------------------
void task_B(void* pvParameters)
{
    (void)pvParameters;
    for (;;)
    {
        // do something


2   xEventGroupSetBits(_myEventGroup, BIT_TASK_B_READY);

    }
}

// ------------------------------------------------------------------
void taskC(void* pvParameters)
{
    (void)pvParameters;
    for (;;)
    {
            xEventGroupWaitBits(
            _myEventGroup,


3   BIT_TASK_A_READY | BIT_TASK_B_READY,
```

```
            pdTRUE,
            pdTRUE,
            portMAX_DELAY);
            // do something
    }
}

// -----------------------------------------------------------
void main(void)
{
    // create tasks and event group
```

BIT_TASK_A_READY & BIT_TASK_B_READY,

xEventGroupSetBits(_myEventGroup, BIT_TASK_A_READY | BIT_TASK_B_READY); → BIT_TASK_A_READY

xEventGroupSetBits(_myEventGroup, BIT_TASK_A_READY & BIT_TASK_B_READY); → BIT_TASK_B_READY

Item 5

String length (3 point)

Look at the code below:

```
char* cPt;
cPt = calloc(16, sizeof(char));
```

If cPt is used as a string in c, how many letters can the string then maximal contain? 1 16 ✕

Correct answers:

1 15

Item 6

Object Oriented design in C (5 points)

Where do the following statements about Object Oriented design in C belong? (drag the statements to the correct colum)

For each correct placed item 1 point and -1 for a wrong placed item. 0 point for items not placed.

True	False
1 Header files can be used as interfaces. ✓	2
Static can be used to implement local variables in module/file scope. ✓	When an object is no longer needed the memory it uses can be reused by setting the pointer to the object to NULL. ✓
C functions can be used to implement "methods" by taking a pointer to the "objects" data as argument. ✓	Since C does not support classes an object oriented design cannot be implemented. ✓

Item 7

FreeRTOS task states (5 points)

Select the right option.
For each statement: a correct answer 1 point.

If a running task is preempted because a high priority task has become available, what state will the task be in when the high priority task starts executing?

1 Running ✕

If a running task (with highest priority) call:

```
xBytesSent = xMessageBufferSend(xMessageBuffer, (void*)pt_strToSend,
    strlen(pt_strToSend) + 1,
    0);
```

and there is enough space, what state will the task be in after the statement?

2 Running ✓

If a task (with highest priority) is in running state and calls:

```
xQueueReceive(xQueue1, &r, portMAX_DELAY);
```

and there is an element in the queue, what state will the task be in after the statement?

3 Blocked ✕

If a running task call:

```
xSemaphoreTake(xSemaphore1, 0);
```

and the semaphore is not taken what state will the task be in after the statement?

4 Blocked ✕

If a running task A calls:

```
xSemaphoreGive(xSemaphore1);
```

and a task B with higher priority is in blocked state waiting for the semaphore to be given.

What state will A be in after the call?

5 Ready ✓

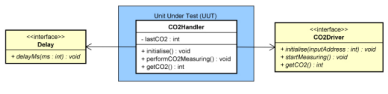
:: Running :: Executing :: Deleted :: Sleeping :: Waiting :: Blocked :: Dormant :: Ready :: Suspended

Correct answers:

1 Ready 2 Running 3 Running 4 Running 5 Ready

Item 8

This class diagram shows a part of a design



To test only the CO2Handler the Delay and CO2Driver must be mocked out using Fake Function Framework (FFF).

Partial points are given.

The source code for the CO2Handler (UUT) can be seen here

```
#include "CO2Handler.h"
#include "CO2Driver.h"
#include "delay.h"

static int lastCO2;

void co2Handler_initialise() {
```

```
    co2Driver_initialise(1000);
}

void co2Handler_performCO2Measuring() {
    co2Driver_startMeasuring();
    delay_delayMs(200);
    lastCO2 = c02Driver_getCO2();
}

int co2Handler_getCO2() {
    return lastCO2;
}
```

As it can be seen is it assumed that the driver is initialised with an inputAddress set to 1000.

Most of the GoogleTest code using FFF is listed here.

Fill in the missing code snippets to make the test code complete.

```
#include "gtest/gtest.h"
#include "../fff.h"

DEFINE_FFF_GLOBALS

extern "C" {
#include <CO2Handler.h>
#include <CO2Driver.h>
#include <delay.h>
}

// Create mocks for Delay
FAKE_VOID_FUNC(delay_delayMs, int);

// Create mocks for CO2Driver
FAKE_VOID_FUNC(co2Driver_initialise, int);
FAKE_VOID_FUNC(co2Driver_startMeasuring);
// fill in the correct mock for co2Driver_getCO2 function
```

1 `FAKE_VALUE_FUNC(int, co2Driver_getCO2);` ✓

```
class CO2HandlerTest : public ::testing::Test {
protected:
    void SetUp() override {
        RESET_FAKE(delay_delayMs);
        RESET_FAKE(co2Driver_initialise);
        RESET_FAKE(co2Driver_startMeasuring);
        // Fill in the missing setup code
```

2 `RESET_FAKE(co2Driver_getCO2);` `FFF_RESET_HISTORY();` ✓

```
    }

    void TearDown() override {}
};

TEST_F(CO2HandlerTest, initialiseIsCallingDriverInitialiseCorrect) {
    // Arrange
    // Act
    co2Handler_initialise();

    // Assert
    ASSERT_EQ(1, co2Driver_initialise_fake.call_count);
    ASSERT_EQ(1000, co2Driver_initialise_fake.arg0_val);
}

TEST_F(CO2HandlerTest, performCO2MeasuringIsCallingDriverCorrect) {
    // Arrange

    // Act
    co2Handler_performCO2Measuring();

    // Assert
    ASSERT_EQ(1, co2Driver_startMeasuring_fake.call_count);
    ASSERT_EQ(1, delay_delayMs_fake.call_count);
    ASSERT_EQ(1, co2Driver_getCO2_fake.call_count);
}

TEST_F(CO2HandlerTest, returnsCorrectMeasuredValue) {
    // Arrange
    co2Driver_getCO2_fake.return_val = 500;

    // Act
    co2Handler_performCO2Measuring();

    // Assert
    // Fill in the correct Assertion
```

3 `ASSERT_EQ(500, co2Handler_getCO2());` ✓

```
}
```

`FAKE_VOID_FUNC(co2Driver_getCO2, int);`

`ASSERT_EQ(1, co2Handler_getCO2_fake.call_count);`

`ASSERT_NE(500, co2Handler_getCO2());`

`ASSERT_EQ(1, co2Driver_getCO2_fake.call_count);`

`FAKE_VALUE_FUNC(co2Driver_getCO2, int);`

`RESET_FAKE(co2Driver_getCO2);`

`FAKE_VOID_FUNC(int, co2Driver_getCO2);` `FFF_RESET_HISTORY();`

`ASSERT_EQ(500, co2Driver_getCO2());`

Item 9

Abstract Data Types (5 points)

Where do the following statements about Abstract Data Types belong? (drag the statements to the correct colum)

For each correct placed item 1 point and -1 for a wrong placed item. 0 point for items not placed.

True	False
1	2
In C a Car can be implemented as an Abstract Data Type. ✓	Abstract Data Types cannot be implemented in C ✓
A list can only be implemented as a linked list in C. ✗	Abstract Data Types are used to hide the implementation details. ✗
All "public" functions used to implement an Abstract Data Type must be listed in the header file. ✓	

Correct answers:

1 Abstract Data Types are used to hide the implementation details.
All "public" functions used to implement an Abstract Data Type must be listed in the header file.
In C a Car can be implemented as an Abstract Data Type.

2 Abstract Data Types cannot be implemented in C
A list can only be implemented as a linked list in C.

Item 10

Priority inversion (5 points)

In one of the code snippets below there is a risk of a priority inversion. Select it.

Task A has the highest priority, task B the second highest priority and task C the lowest.

Right answer 5 points.

✗ Code

1

```
// ------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {
```

```
        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskC(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void main(void)
{
    xSemaphore = xSemaphoreCreateBinary();
    xSemaphoreGive(xSemaphore);
    // create tasks and start schedular
```

```
// ------------------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        // do something
    }
}
// ------------------------------------------------------------------------------
void taskC(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void main(void)
{
    xSemaphore = xSemaphoreCreateBinary();
    xSemaphoreGive(xSemaphore);
    // create tasks and start schedular
```

```
// ------------------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        // do something
    }
}
// ------------------------------------------------------------------------------
void taskC(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void main(void)
{
    xSemaphore = xSemaphoreCreateMutex();
    // create tasks and start schedular
```

```
// ------------------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore2, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreTake(xSemaphore2, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore3, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskC(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore3, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore1, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void main(void)
{
    xSemaphore1 = xSemaphoreCreateBinary();
    xSemaphore2 = xSemaphoreCreateBinary();
    xSemaphore3 = xSemaphoreCreateBinary();
    xSemaphoreGive(xSemaphore1);
    // create tasks and start schedular
```

```
// ------------------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore1, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreTake(xSemaphore2, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore2, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskC(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore3, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore3, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void main(void)
{
    xSemaphore1 = xSemaphoreCreateBinary();
    xSemaphore2 = xSemaphoreCreateBinary();
    xSemaphore3 = xSemaphoreCreateBinary();
    xSemaphoreGive(xSemaphore1);
    // create tasks and start schedular
```

Correct answers:

1

```
// ------------------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {

        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        // do something
```

```
}
//-----------------------------------------------------------
void taskC(void* pvParameters)
{
    for (;;)
    {
        // do something
        xSemaphoreTake(xSemaphore, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore, portMAX_DELAY);
    }
}
//-----------------------------------------------------------
void main(void)
{
    xSemaphore = xSemaphoreCreateBinary();
    xSemaphoreGive(xSemaphore);
    // create tasks and start scheduler
```

Item 11

Memory 2 (5 Points)

Look at the following code:

```
int x;
int* ipt;
x = 24;
ipt = &x;
*ipt = 42;
printf("*ipt: %d. x: %d\n",*ipt, x);
```

Are there any potential problems with this code?
Right answer 5 points.

- Yes, it will overwrite address 24 (depending on the compiler it may or may not compile).
- Yes, it will print the address of ipt not what ipt is pointing to (depending on the compiler it may or may not compile).
- ✓ No, it will run and print "*ipt: 42, x: 42"
- ✗ No, it will run and print "*ipt: 42, x: 24"
- No, it will run and print "*ipt: 24, x: 42"
- Yes, it will print the address of x not the contents of x (depending on the compiler it may or may not compile).

Item 12

Structs (5 points)

The code below is unfinished. A printf statement needs to be completed.
There are several options. Some will compile and some not.

Place the code snippets in the correct column.

Each correct answer gives one point.

```
struct student_s
{
    int id;
    char* name;
};
struct student_s viaStudent[2];
struct student_s* stp = &viaStudent[0];

viaStudent[0].id = 123456;
viaStudent[0].name = "Peter Pan";

viaStudent[1].id = 159263;
viaStudent[1].name = "Marci Pan";

// Print out the name to the console
// Replace this line with a printf statement
```

Will compile	Will not compile
1 printf("Name: %s\n", viaStudent[0].name); ✓	2 printf("Name: %s\n", stp.name); ✓
printf("Name: %s\n", stp->name); ✓	printf("Name: %s\n", (viaStudent + 1).name); ✓
printf("Name: %s\n", (stp + 1)->name); ✓	

Item 13

Pointers 4 (2 points)

Consider the C program below:

```
void pointers4()
{
    int  x = 25;
    int* y = &x;
    int* z = calloc(1, sizeof(int));

    if (NULL != z)
    {
        *z = *y;
        *y = 11;
        printf("1: %d, 2: %d, 3: %d\n", x, *y, *z);
        free(z);
    }
}
```

Which values are printed to the console?
(one number in each box)
All answers must be correct in order to score points.

1: 1 25 ✗ 2: 2 11 ✓ 3: 3 11 ✗

Correct answers:

1 11 3 25

Item 14

C standard library (5 points)

The C standard Library functions and definitions are declared in header files.

Place the shown items under the appropriate header file.

Hint: some items does not belong to any of the shown headers. Just leave them.

For each correct placed item 0,5 point.

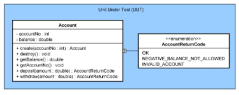
stdint.h	string.h	stdio.h	stdlib.h	math.h
1 uint8_t ✓	2 strncpy ✓	3 getchar ✓	4 free ✓	5 pow ✓
int16_t ✓	strlen ✓	sprintf ✓		
		EOF ✓		
		puts ✓		

unsigned int | string | double

Item 15

Test cases (6 points)

This design is implemented below.



Source file Account.h:

```
#pragma once
typedef  struct account * account_t;
typedef  enum { OK, NEGATIVE_BALANCE_NOT_ALLOWED, INVALID_ACCOUNT }
accountReturnCode_t;

account_t account_create(int accountNo);
double account_getBalance(account_t self);
int account_getAccountNo(account_t self);
accountReturnCode_t account_deposit(account_t self, double amount);
accountReturnCode_t account_withdraw(account_t self, double amount);
```

Source file Account.c:

```
#include <stdlib.h>

#include "Account.h"

struct account {
    int accountNo;
    double balance;
};

account_t account_create(int accountNo) {
    account_t self = malloc(sizeof(struct account));
    if (NULL != self) {
        self->accountNo = accountNo;
        self->balance = 0.0;
    }

    return self;
}

void destroy(account_t self) {
    if (NULL != self) {
        free(self);
    }
}

double account_getBalance(account_t self) {
    return self->balance;
}

int account_getAccountNo(account_t self)
{
    return self->accountNo;
}

accountReturnCode_t account_deposit(account_t self, double amount) {
    self->balance += amount;
    return OK;
}

accountReturnCode_t account_withdraw(account_t self, double amount) {
    if (self->balance >= amount) {
        self->balance -= amount;
        return OK;
    }

    return NEGATIVE_BALANCE_NOT_ALLOWED;
}
```

Assuming the implementation is correct which of the following tests will pass and which will fail?

Drag each test cases to either pass or fail. For each test cases: a correct answer 1 point, no answer 0 point and -1 for a wrong answer.

Will pass

1

```
TEST(AccountTest, balanceEmptyWhenCreated) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);

  EXPECT_DOUBLE_EQ(0.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, possibleToWithdrawAmount) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    account_deposit(uut, 100.0);

    EXPECT_EQ(OK, account_withdraw(uut, 100.0));
    EXPECT_DOUBLE_EQ(0.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, correctBalanceAfterWithdraw) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    account_deposit(uut, 100.0);

    EXPECT_NE(OK, account_withdraw(uut, 50.0));
    EXPECT_DOUBLE_EQ(50.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, possibleToDepositMoreAmounts) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    account_deposit(uut, 100.0);
    account_deposit(uut, 100.0);

    EXPECT_DOUBLE_EQ(200.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, impossibleToWithdrawFromNewAccount) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    EXPECT_EQ(NEGATIVE_BALANCE_NOT_ALLOWED, account_withdraw(uut, 1.0));
}
```

Will fail

2

```
TEST(AccountTest, possibleToDepositAmount) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    accountReturnCode_t rc = account_deposit(uut, 100.0);

    EXPECT_EQ(OK, rc);
    EXPECT_DOUBLE_EQ(1000.0, account_getBalance(uut));
}
```

Correct answers:

1

```
TEST(AccountTest, balanceEmptyWhenCreated) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);

  EXPECT_DOUBLE_EQ(0.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, impossibleToWithdrawFromNewAccount) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);

    EXPECT_EQ(NEGATIVE_BALANCE_NOT_ALLOWED, account_withdraw(uut, 1.0));
}
```

```
TEST(AccountTest, possibleToDepositMoreAmounts) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    account_deposit(uut, 100.0);
    account_deposit(uut, 100.0);

    EXPECT_DOUBLE_EQ(200.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, possibleToWithdrawAmount) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    account_deposit(uut, 100.0);

    EXPECT_EQ(OK, account_withdraw(uut, 100.0));
    EXPECT_DOUBLE_EQ(0.0, account_getBalance(uut));
}
```

2

```
TEST(AccountTest, possibleToDepositAmount) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    accountReturnCode_t rc = account_deposit(uut, 100.0);

    EXPECT_EQ(OK, rc);
    EXPECT_DOUBLE_EQ(1000.0, account_getBalance(uut));
}
```

```
TEST(AccountTest, correctBalanceAfterWithdraw) {
    int accountNo = 1000;
    account_t uut = account_create(accountNo);
    account_deposit(uut, 100.0);

    EXPECT_NE(OK, account_withdraw(uut, 50.0));
    EXPECT_DOUBLE_EQ(50.0, account_getBalance(uut));
}
```

Item 16

Format specifier (4 points)

Select the correct format specifier for printf.
For each drop down menu a correct answer give 1 point.

```
unsigned int a = 22;

int x = 7;

float s = 29.95;

double d = 22.0 / 7;

char* t = "single letter";

printf("Pi is approximately 1 %u /%d = 2 %f \n", a, x, d);

printf("Pizza slice: 3 %.2f \n", s);

printf("Don\'t use 4 %s variable names!\n", t);
```

Item 17

Operating system (5 points)

Which statements about real time operating systems are true? Fill out the table.
For each statement: a correct answer 1 point, no answer 0 point and -1 for a wrong answer.

Deadlocks may happen when there are three tasks and one ressource protected by a semaphore.	1 False
Priority inversion may happen when there are two tasks and two ressources each protected by a semaphore.	2 False
A task holding a ressource protected by a mutex cannot be preempted.	3 False
Semaphores and Mutexes are used to prevent deadlocks	4 True

Semaphores and Mutexes are used to prevent deadlocks.

Two tasks cannot share a queue.

5 True ✕

True | False

Correct answers:

1 False | 2 False | 3 False | 4 False | 5 False

Item 18

Pointers 1 (2 Points)

What will this code snipped print out on the console?

```
void bar()
{
    char myWorld[] = {'M', 'y', ' ', 'w', 'o', 'r', 'l', 'd'};
    printf("1: %c, 2: %c, 3: %c", myWorld[0], *(myWorld + 4), *myWorld);
}
```

One letter in each box.
All answers must be correct in order to score points.

1: 1 M ✓ 2: 2 o ✓ 3: 3 M ✓

Item 19

Test cases and ZOMBIES (5 points)

The test cases below are from a TDD project following the principles of ZOMBIES.

Here is a part of the class diagram for the system

Drag each test cases to the letter that best describe it's scenario. Each code snippets placed correct gives 1.25 points.

Z
1
```
TEST_F(TourTest, initialisedCorrect) {
    // Arrange
    // Act
    tour_initialise();

    // Assert
    ASSERT_EQ(0, tour_remainingWaypoints());
}
```

O
2
```
TEST_F(TourTest, canAcceptWaypoint) {
    // Arrange
    // Act
    tour_initialise();
    tour_addWaypoint(1, 2, 3);

    // Assert
    ASSERT_EQ(1, tour_remainingWaypoints());
}
```

M
3
```
TEST_F(TourTest, canAcceptWaypoints) {
    // Arrange
    // Act
    tour_initialise();
    tour_addWaypoint(1, 2, 3);
    tour_addWaypoint(4, 5, 6);

    // Assert
    ASSERT_EQ(2, tour_remainingWaypoints());
}
```

B
4
```
TEST_F(TourTest, willNotStartCarWithEmptyTour) {
    // Arrange
    // Act
    tour_initialise();
    tour_startTour();

    // Assert
    ASSERT_EQ(0, car_start_fake.call_count);
}
```

I
5

E
6

S
7

Correct answers:

1
```
TEST_F(TourTest, initialisedCorrect) {
    // Arrange
    // Act
    tour_initialise();

    // Assert
    ASSERT_EQ(0, tour_remainingWaypoints());
}
```

```
TEST_F(TourTest, willNotStartCarWithEmptyTour) {
    // Arrange
    // Act
    tour_initialise();
    tour_startTour();

    // Assert
    ASSERT_EQ(0, car_start_fake.call_count);
}
```

2
```
TEST_F(TourTest, canAcceptWaypoint) {
    // Arrange
    // Act
    tour_initialise();
    tour_addWaypoint(1, 2, 3);

    // Assert
    ASSERT_EQ(1, tour_remainingWaypoints());
}
```

3
```
TEST_F(TourTest, canAcceptWaypoints) {
    // Arrange
    // Act
    tour_initialise();
    tour_addWaypoint(1, 2, 3);
    tour_addWaypoint(4, 5, 6);

    // Assert
    ASSERT_EQ(2, tour_remainingWaypoints());
}
```

4 No correct answers set

5 No correct answers set

6 No correct answers set

7 No correct answers set

Item 20

Task synchronization (5 points)

Below are some programs, choose the one where taskA signals taskB to execute.

Right answer 5 points

code

1
```
xSemaphoreHandle xSemaphore1;
// ------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {
        // do something
        xSemaphoreGive(xSemaphore1, portMAX_DELAY);
    }
}
// ------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
```

```
    {
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
        // do something
    }
}
// ------------------------------------------------------------------
void main(void)
{
    xSemaphore1 = xSemaphoreCreateBinary();
    // create tasks and start schedular
```

```
xSemaphoreHandle xSemaphore1;
// ------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {
        // do something
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
        // do something
    }
}
// ------------------------------------------------------------------
void main(void)
{
    xSemaphore1 = xSemaphoreCreateBinary();
    // create tasks and start schedular
```

```
xSemaphoreHandle xSemaphore1;
xSemaphoreHandle xSemaphore2;
// ------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore2);
    }
}
// ------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreTake(xSemaphore1, portMAX_DELAY);
        // do something
        xSemaphoreGive(xSemaphore2);
    }
}
// ------------------------------------------------------------------
void main(void)
{
    xSemaphore1 = xSemaphoreCreateBinary();
    xSemaphore2 = xSemaphoreCreateBinary();
    xSemaphoreGive(xSemaphore1);
    // create tasks and start schedular
```

```
xSemaphoreHandle xSemaphore1;
// ------------------------------------------------------------------
void taskA(void* pvParameters)
{
    for (;;)
    {
        // do something
        xSemaphoreGive(xSemaphore1, portMAX_DELAY);
    }
}
// ------------------------------------------------------------------
void taskB(void* pvParameters)
{
    for (;;)
    {
        xSemaphoreGive(xSemaphore1, portMAX_DELAY);
        // do something
    }
}
// ------------------------------------------------------------------
void main(void)
{
    xSemaphore1 = xSemaphoreCreateBinary();
    // create tasks and start schedular
```

Item 21

Pointers 3 (3 points)

Consider the C program below:

```
int pointers3()
{
int  x = 16;
int* y = calloc(1, sizeof(int));
int* z = &x;

    if (NULL != y)
    {
        *y = 32;
        x = x + *y + *z;
        free(y);
    }
    return x;
}
```

What value is returned?
(one number in each box)
Both answers must be correct in order to score points.

1. If the allocation is successful: 1 — 64 ✓
2. If the allocation fails: 2 — 16 ✓

Item 22

Memory 1 (5 Points)

The following is implemented in a C-source file:

```
typedef struct item {
    int32_t serialNr;
    int16_t cost;
}item_st;

//Missing line

void bar(item_pt i) {
    printf("serialNr: %d.\ncost: %d.", i->serialNr, i->cost);
}

int main() {
    item_pt i = (item_pt)malloc(sizeof(item_pt));
    if (i) {
        i->serialNr = 42;
        i->cost = -12;
    }
    bar(i);

    return 0;
}
```

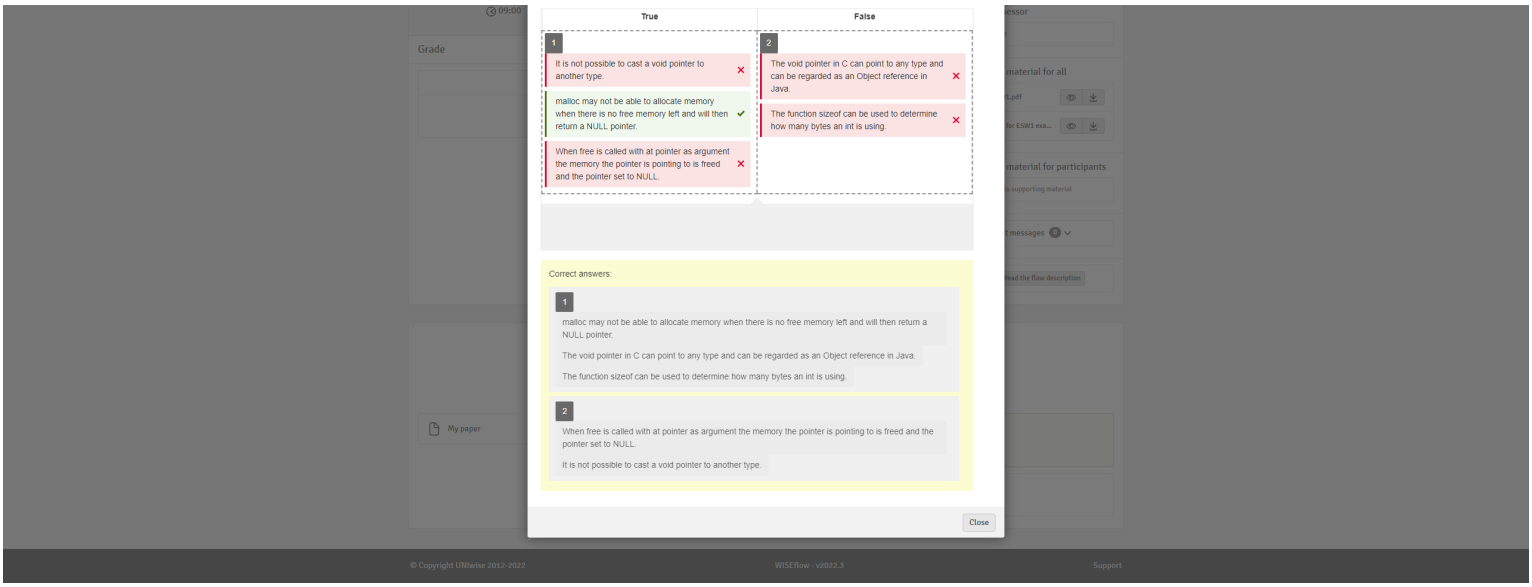
Choose the right line so the program will compile and run.
Right answer 5 points.

- item_st* item_pt;
- item_st* i;
- typedef item_st item_pt;
- typedef item_pt item_st;*
- ✓ typedef item_st* item_pt;

Item 23

Memory 3 (5 points)

Where do the following statements about memory belong? (drag the statements to the correct colum)
For each correct placed item 1 point and -1 for a wrong placed item. 0 point for items not placed.


09:00
Grade
My paper
True
False
1
It is not possible to cast a void pointer to another type.
malloc may not be able to allocate memory when there is no free memory left and will then return a NULL pointer.
When free is called with at pointer as argument the memory the pointer is pointing to is freed and the pointer set to NULL.
2
The void pointer in C can point to any type and can be regarded as an Object reference in Java.
The function sizeof can be used to determine how many bytes an int is using.
Correct answers:
1
malloc may not be able to allocate memory when there is no free memory left and will then return a NULL pointer.
The void pointer in C can point to any type and can be regarded as an Object reference in Java.
The function sizeof can be used to determine how many bytes an int is using.
2
When free is called with at pointer as argument the memory the pointer is pointing to is freed and the pointer set to NULL.
It is not possible to cast a void pointer to another type.
Close
material for all
material for participants
messages
© Copyright UNIwise 2012-2022
WISEflow - v2022.3
Support