Pregunta 1

0 de 2 puntos

Escribir una función para realizar el producto escalar de 2 vectores.
La función debe recibir ambos vectores (que poseen números reales) y el largo de ambos. Debe devolver el resultado del producto escalar. Debe funcionar para distintos largos de vectores, es decir, el largo no debe ser constante.
Ayuda: el producto escalar de *v1=[1,2,3]* y *v2=[4,5,6]* es *1*4+2*5+3*6=32*

Respuesta seleccionada:

```
#include <stdio.h>
#define N 3
double productoEscalar(double v1, double v2, int n1, int n2);
int main()
{
    double respuesta;
    double v1[N] = {1, 2, 3};
    double v2[N] = {4, 5, 6};
    int n1 = sizeof(v1)/sizeof(v1[0]);
    int n2 = sizeof(v2)/sizeof(v2[0]);

    respuesta = productoEscalar(v1, v2, n1, n2);
}
double productoEscalar(v1, v2, n1, n2)
{
    int i;
    double resp = 0;
    for(i=0; i<n1; i++)
    {
        resp = resp + v1[i]*v2[i];
    }
    return resp;
}
```

Comentarios para respuesta: NO comenta. -0,5

No compila: faltan tipos de datos en definición de función. -1

Argumento n2 innecesario: nunca lo usa. -0,5

Corrigiendo esto funciona OK

Pregunta 2

0,5 de 2 puntos

De las siguientes afirmaciones, algunas son verdaderas y otras falsas.
Corregir las afirmaciones falsas, modificando la parte subrayada, para que pasen a ser verdaderas.
Se penaliza modificar afirmaciones verdaderas.

Sea `arr` un arreglo de 4 `int` y sea `pun` un puntero a `int` que apunta al segundo elemento de `arr`.

a. El tamaño de `pun` depende del tipo de dato al que apunta
b. Hacer `*arr=5;` genera un error
c. `*p++;` incrementa el valor del segundo elemento del arreglo
d. `arr + 1` no es un *lvalue*
e. `arr++;` no es una operación válida
f. `&p` devuelve la dirección del segundo elemento de `arr`
g. `(p+1)=0;` le asigna 0 al último elemento de `arr`
h. Para decrementar el primer elemento de `arr` mediante `pun` se debe hacer `--p[-1];`

Respuesta seleccionada: Sea arr un arregle de 4 int y sea p un puntero a int que apunta al segundo elemento de arr.

```
int arr[4];
int *p = &arr[1];
```

a.
El tamaño de `p` depende del tipo de dato al que apunta
Verdadero: porque tiene que saber cuantas cantidades de byte tiene que recorrer en la memoria

b.
Hacer `*arr=5;` genera_ un error
Verdadero:

c.
`*p++;` incrementa el valor del segundo elemento del arreglo
Verdadero:
Basandome en la tabla de precedencia. El " * " esta en el mismo nivel que el "++", por lo que tiene importancia de derecha hacia izquierda.
Entonces esto seria as: `*(p++)`. Pero como es un posincremento. Primero *(p), lo desreferencia y luego le incrementa

d.
`arr + 1` no es un *lvalue*
Verdadero: el array es solo un puntero de solo lentura

e.
`arr++;` no es una operación válida
Verdadero: Al ser un array de lectura, no se le permite asignacion

f.
`&p` devuelve la dirección del segundo elemento de `arr`
Falso: porque el puntero apunta a la direccion de memoria del segundo elemento del array. Y si queres la direccion de memoria del segundo elemento del array es simplemente "p".
&p devuelve la direccion de donde fue creado el puntero a int.
&p devuelve la dirección del puntero p

g.
`(p+1)=0;` le asigna 0 al último elemento de `arr`
Falso: le asigna un 0 a la direccion de memoria donde esta contenido el tercer elemento de arr

`(p+1)=0;` le asigna 0 a la direccion de memoria donde esta contenido el tercer elemento de arr

h.
Para decrementar el primer elemento de `arr` mediante `p` se debe hacer `--p[-1];`
Verdadero: Por precedencia veo el p[-1], toma el valor. Luego con el predecremento, le resta 1.

Comentarios para respuesta: a. FALSA El tamaño de pun depende DE LA ARQUITECTURA/TAMAÑO DE MEMORIA -0,5

b. FALSA Hacer *arr=5; ASIGNA 5 AL PRIMER ELEMENTO DEL ARREGLO -0,5

c. FALSA *p++; incrementa el valor del PUNTERO -0,5

Pregunta 3 0,5 de 1,5 puntos

Explicar la diferencia de utilizar el condicional `if - else` respecto del `#if - #else`.
Dar un ejemplo concreto (incluyendo código) dónde convenga utilizar cada uno.

Respuesta seleccionada: La diferencia entre `if - else` y `#if - #else`. Son los tiempo de ejecución.
La primera, se ejecuta en tiempo de ejecucion, o sea cuando ejecutas el archivo .exe
En cambio las directivas del segundo, se realizan en la etapa de preprocesador y se ejecutan ahi. Quiere decir que cuando ejecutes el .exe, esas lineas ya estan implementadas.

Comentarios para respuesta: Explicación OK.

Faltan ejemplos.

Pregunta 4 0 de 1,5 puntos

Indicar qué imprime el siguiente código. **Justificar la respuesta**.

```
#include <stdio.h>
int main(void)
{
    int var = -11.7;
    printf("%010f\n", (double) (var--/3));
    return(0);
}
```

Respuesta seleccionada:

```
#include <stdio.h>
int main(void)
{
1    int var = -11.7;
2    printf("%010f\n", (double) (var--/3));
3    return(0);
}
```

```
Linea 1:
Me castea el var a -11. Quiere decir que var = -11
Linea 2:
El casteo a (double) tiene el mismo nivel de precedencia que el "--". Por lo que su importancia va a derecha a izquierda
La operacion var-- es posdecremento, por lo que primero hace el casteo a (double). Una vez hecho el casteo a double hace el decremento.
(double) var-- = -10.0000
var/3 = -10.0000/3 = -3.0000
la parte de "%010f\n": quiere decir que quiero imprimir un solo 0 delante y luego que en total haya 10 digitos(incluyendo el -). Luego indica que imprime una variable de
tipo float/double y luego hace un salto de linea

Imprime
-03.000000
```

Comentarios para respuesta:

Respuesta OK pero mal justificado (varios errores que se cancelan entre sí).

1) La división se hace antes que el casteo (el paréntesis cambia la precedencia).

2) Decrementar -11 da -12, no -10

3) double/int devuelve un double, no un int (-10.0000/3 = -3.3333)

Pregunta 5

0,5 de 3 puntos

Escribir la función `seno` que calcule el seno de x mediante la *serie de Taylor*:

$$seno(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

La función debe recibir el valor de x (en radianes) y devolver el valor calculado.
La cantidad de términos a utilizar está definida en la constante entera `N_TERM` (escribir dicha definición).
Se permiten escribir subfunciones para resolver el ejercicio.