FUNDAMENTOS DE PROGRAMACION	MODELO 3	*CONV. ORD. 15/16*
GRADO DE INFORMÁTICA y GRADO DE TECNOLOGÍAS DE LA INFORMACIÓN-ETSII-UNED		
MATERIAL AUTORIZADO: **NINGUNO**		DURACIÓN: **2 HORAS**

INSTRUCCIONES:	**1) El test debe ser contestado en la hoja de marcas. Sólo una de las cuatro respuestas posibles de cada pregunta es correcta.** **2) El test es eliminatorio y aporta un 30% de la nota final. Son necesarias 8 preguntas correctas (6 con las prácticas aprobadas) para que se corrija el ejercicio.** **3) Cada respuesta correcta 1 pto. Respuesta incorrecta, doble o en blanco: 0 pto.** **4) Puede quedarse, si lo desea, con esta hoja de examen.**

CUESTIONES DE TEST

1. En C±, la importación de librerías se realiza con:

A. #import
B. #include
C. #require
D. #introduce

2. La eficiencia de un programa equivale a medir:

A. los recursos de tiempo para ejecutarse y la memoria de datos que ocupa.
B. la corrección parcial y total del programa.
C. la estructura imperativa del programa
D. los recursos de sentencias y comentarios totales del programa

3.- El siguiente código:

```
int procedimiento(int i) {
  j = i * j;
  i = j + 1;
}
..
i = 2;
j = 3;
procedimiento(i);
printf("%d %d", j, i);
```

A. produce un error en tiempo de ejecución
B. imprime 6 2
C. imprime 6 7
D. produce un error en tiempo de compilación

4.- En el lenguaje C ±, la declaración de un tipo struct debe tener al menos…

A. un campo o subprograma privado
B. un campo o subprograma público o privado
C. un campo público o privado
D. un campo o subprograma público

5.- En el lenguaje C ±, el siguiente código:

```
a = true;
try {
    if (a || !a) {
      throw true;
    } else {
      throw false;
    }
} catch (bool a) {
    printf("%d", int(a));
}
```

A. imprime 1
B. produce un error en tiempo de compilación
C. imprime 0
D. produce un error en tiempo de ejecución

6.- Para realizar una estructura de datos no acotada es necesario utilizar:

A. formaciones
B. punteros
C. tipos abstractos de datos (TAD)
D. tablas

7. La ejecución del siguiente código:

```
i = 6; j = 0;
while (i>j) {
  switch ((j)%2){
    case 0: j=j+1; break;
    case 1: i--; j=i+1; break;
  }
  printf ("%d, %d; ", j, i);
```

A. imprime 1, 6;
B. no imprime nada
C. imprime 1, 6; 2, 5; 3, 5; 4, 4;
D. imprime 1, 6; 6, 5;

8. La sentencia que utilizamos en C+/- en una selección múltiple por casos para realizar las acciones alternativas a cualquier otra condición es:

A. else
B. break
C. default
D. switch

9. El paso de argumentos de tipo formación….

A. es por defecto por valor
B. debe estar precedido del símbolo &
C. es por defecto por referencia
D. debe estar precedido por const

10. En C+/-, cómo se realizaría la declaración de un tipo para representar un punto en el plano:

A. typedef record TipoPunto { float x; float y};
B. typedef TipoPunto = { float x; float y};
C. typedef TipoPunto { float x; float y};
D. typedef struct TipoPunto { float x; float y};

EJERCICIO DE PROGRAMACIÓN

Realizar un tipo abstracto de datos Dispensador para almacenar hasta 10 productos diferentes y el total de recaudación. Los datos de cada producto son el código de producto (entero entre 1 y 1000), las unidades disponibles (máximo 20 unidades) y el precio por unidad. Las operaciones son: IniciarDispensador inicia los 10 productos con el código 0 y pone la recaudación a cero, CargarProducto que aumenta, si hay hueco en el código de producto dado, las unidades cargadas y sino crea un nuevo producto en el dispensador cuando hay hueco y modifica/inicia el precio unitario, ComprarProducto que aumenta la recaudación en su precio, disminuye en una unidad el producto de código introducido y si es el último se pone el código 0.