RELLENE EN ESTA HOJA Y EN LA HOJA DE LECTURA ÓPTICA LOS SIGUIENTES DATOS:

Apellido**s:**.....Tlfno.**:**.....

Nombre**:**.....D.N.I.**:**.....

Código Carrera: **40 (Sist.)** Código Asignatura: **103** Convocatoria: **Septiembre 1ªPP**

41(Gest.) Semana: **1ª**

Tipo de Examen: **A**

ð El **test** debe ser contestado **en la hoja de lectura óptic**a. Sólo una de las cuatro respuestas posibles de cada pregunta es correcta.

ð El test es eliminatorio y aporta un 40% de la nota final. Son necesarias 7 respuestas correctas para que se corrija el ejercicio.

ð La solución del ejercicio se realizará en el reverso de esta hoja. **No se corregirán hojas auxiliare**s.

ENTREGUE ÚNICAMENTE ESTA HOJA Y LA HOJA DE LECTURA ÓPTICA sin grapar

TEST (cada respuesta correcta: 1punto; respuesta incorrecta o en blanco: 0 puntos)

1. Dado el siguiente fragmento de código:

CASE j OF

enero..junio:INC(j);|

diciembre..julio:DEC(j);

ELSE

END;

A.− la variable j tiene que ser de tipo ordinal para ser correcto

B.− es necesaria alguna sentencia tras ELSE

C.− la sentencia ELSE no se puede usar dentro de una instrucción CASE

D.− los rangos establecidos son incorrectos

2. Para que en Modula−2, la siguiente definición de tipo sea la de una tabla:

TYPE TipVect= ARRAY TipInd OF TipElem

A.− TipInd debe ser un tipo ordinal definido por el usuario y TipElem de cualquier

tipo

B.− TipInd debe ser cualquier tipo predefinido y TipElem RECORD

C.− TipInd debe ser cualquier tipo ordinal y TipElem RECORD

D.− TipInd debe ser RECORD y TipElem de cualquier tipo

3. Dado el siguiente procedimiento:

PROCEDURE proc(a,b:INTEGER);

VAR aux:INTEGER;

BEGIN

aux:=a+b+c;

c:=aux;

END proc;

Para que fuese puro:

A.− Todas las variables, incluida aux, deben pasarse por referencia

B.− Es ya un procedimiento puro

C.− bastaría con pasar las variables a y b por referencia

D.− la variable c se debería pasar por referencia

4. Sabiendo que en Modula−2 existe el tipo predefinido BOOLEAN:

TYPE BOOLEAN=(FALSE, TRUE)

Se puede afirmar:

A.− ORD(FALSE)=0

B.− ORD(FALSE)=1

C.− El tipo BOOLEAN no es un tipo ordinal

D.− Con los tipos predefinidos no se puede utilizar la ORD

5. La compilación segura:

A.− Mejora la reutilización

B.− Produce un programa objeto más eficiente

C.− Tiene como objetivo comprobar la compatibilidad de tipos

D.− Necesita un modulo de definición

6. Dado el siguiente fragmento de código:

TYPE t1=RECORD c1, c2:REAL; END;

t2=POINTER TO REAL;

t3=POINTER TO t1;

VAR a:t1; b:t2; c:t3;

...

NEW(b); NEW(c);

La asignación correcta es:

A.− c^.c1:=a.c2;

B.− b:= c^.c1;

C.− a:=c;

D.− c.c1:=a.c2;

7. Dado la siguiente declaración de variables:

VAR c1:T1; c2:T2;

Después de ejecutar

Leer(c1); Leer(c2);

las variables c1 y c2 toman los valores esto y aquello respectivamente.

A la vista del resultado, el procedimiento Leer tendrá como argumento:

A.− (VAR a: ARRAY OF CHAR)

B.− (a:T1,T2)

C.− (VAR a:T1,T2)

D.− (a: ARRAY OF CHAR)

8. Después de ejecutar el siguiente fragmento de código:

VAR pt1, pt2, pt3: POINTER TO REAL;

...

NEW(pt1); pt1^:=3.0; pt2:=pt1; pt3:=pt2;

DISPOSE(pt2);

A.− Sólo se libera pt3

B.− Se liberan los punteros pt1, pt2 y pt3

C.− Sólo se libera pt3 y pt2

D.− Se produce un error, y no se libera la memoria

9. La definición completa de un tipo opaco que se declara en un módulo de

definición debe completarse en :

A.− La declaración también se hace en el módulo de implementación

B.− El módulo principal del programa

C.− El propio módulo de definición

D.− El módulo de implementación correspondiente

10. Dado el siguiente procedimiento:

PROCEDURE

Calcular (VAR A: INTEGER; B: INTEGER) : INTEGER;

BEGIN

A := B MOD 2; B := A * A + 3 * B − 6; RETURN A+B

END Calcular;

El valor de X tras ejecutar X:=4; X:=Calcular(X,X); será:

A.− 4

B.− 0

C.− 6

D.− 12

EJERCICIO DE PROGRAMACIÓN (10 puntos)

Construir un dato encapsulado que sea una tabla de 50 elementos, en la que se almacenan los datos de una persona: nombre,

apellido1, apellido2, dirección y teléfono. Las operaciones serán añadir un elemento, eliminar un elemento y un procedimiento de

búsqueda selectiva por nombre o por número de teléfono. Si se encuentra la persona en la tabla se devolverá cierto y se mostrará

toda la información disponible de esa persona y si no se devolverá falso.

D.N.I. : **CENTRO ASOCIAD**O:

INGENIERIA TECNICA en INFORMATICA de GESTION y SISTEMAS

ASIGNATURA : PROGRAMACION I ORIGINAL

CODIGO CARRERA: **41** = GESTION CONVOCATORIA: FEBRERO 1ª PP

40 = SISTEMAS SEMANA: 1ª Semana TIPO EXAMEN: **1**

CODIGO ASIGNATURA: **103**

DURACIÓN : 2 horas FECHA: 25−I−1999

MATERIAL AUXILIAR: NINGUNO HORA: 11:30 horas

¡ATENCIÓN! PONGA EL TIPO DE EXAMEN EN LA HOJA DE LECTURA OPTICA.

Contéstese el test en la hoja de lectura óptica.

El test es **ELIMINATORIO** (son necesarias 7 respuestas correctas para pasarlo), y aporta el 40% de la nota final.

Sólo hay una respuesta correcta en cada pregunta.

1. Respecto a la sentencia:

Correcto := calculo(matriz) IN grupo;

Se puede decir en cualquier caso que:

A.− La función calculo devuelve un valor enumerado

B.− calculo es un procedimiento que devuelve como resultado matriz

C.− matriz y grupo son de tipos compatibles

D.− grupo es de tipo BOOLEAN

2. Los metasimbolos son:

A.− Elementos de la notación BNF

B.− Elementos de la programación lógica

C.− Elementos de la programación funcional

D.− Parte del modelo de flujo de datos

3. La reutilización se consigue mediante desarrollo:

A.− Ascendente o descendente

B.− Sólo ascendente

C.− Sólo descendente

D.− Robusto

4. Los tipos opacos:

A.− Son tipos abstractos de datos

B.− Son datos persistentes

C.− Son punteros

D.− Son de acceso secuencial

5. Dado el siguiente fragmento de código:

```
FOR i:=0 TO 1 BY 0.1 DO
WriteString("Esto se escribe 10 veces");
END
```

A.− Se produce un error por incompatibilidad de tipos.

B.− El cuerpo del bucle se ejecuta 10 veces.

C.− La frase se escribe una vez, pero al incrementar el índice se produce un error.

D.− Aunque no hay errores, el cuerpo del bucle nunca se ejecuta.

6. Dado el siguiente fragmento de código:

```
VAR a:CHAR;
TYPE CHAR=SET OF ['1'..'9'];
VAR b:CHAR;


b:=CHAR{'3'..'5'};
```

a:=b;

A.− Se produce un error en la asignación a:=b.

B.− La declaración de las variables es incompatible.

C.− Se produce un error en la asignación b:=CHAR{'3'..'5'}.

D.− Es correcto.

7. A la vista únicamente de la siguiente declaración:

VAR K:POINTER TO SET OF [1..10];

Sólo se puede decir que:

A.− K es de tipo anónimo.

B.− Es incorrecta.

C.− K es un dato encapsulado.

D.− K es de tipo opaco.

8. Dada la siguiente fragmento de código

dato=dias{L};

podemos decir que

A.− dias{L} es un valor constante de tipo conjunto

B.− L puede ser una variable o una constante

C.− dias es el nombre del tipo referencial

D.− dato no tiene tipo

9. Dado el siguiente módulo

DEFINITION MODULE Segundo;

PROCEDURE Previo(VAR dato:tipodato):

END Segundo.

A.− Es erróneo, se necesita la declaración de tipodato

B.− Es correcto, tipodato es un tipo opaco

C.− Es erróneo tipodato es un tipo anónimo

D.− Es erróneo dato debería ser por valor

10. La cabecera del subprograma WriteString para imprimir cadenas de

caracteres podría ser

A.− PROCEDURE WriteString(a:ARRAYOF CHAR);

B.− PROCEDURE WriteString(VAR a:ARRAY[1..1000]OF CHAR);

C.− PROCEDURE WriteString(a:CHAR);

D.− PROCEDURE WriteString(a:POINTER TO CHAR);

EJERCICIO DE PROGRAMACIÓN

En el módulo Juegos, se dispone del tipo abstracto de datos CartaBaraja, que representa una carta de la baraja española. También se dispone de

dos operaciones asociadas: PonerPalo y PonerTriunfo.

PonerPalo, establece el palo de la carta: oros, copas, espadas o bastos.

Por ejemplo: PonerPalo(carta,oros).

PonerTriunfo, establece el triunfo de la carta: as, dos,, sota, caballo o rey.

Por ejemplo: PonerTriunfo(carta,rey)

Se pide crear en el módulo principal una baraja de 40 cartas y la operación Vencer. Una carta vence a otra cuando su triunfo es mayor, excepto

cuando una de las cartas es de la pinta que entonces gana aún cuando su triunfo sea menor. Si dos cartas tienen igual triunfo vence cualquiera

de las dos.

Ejemplo: carta1 es el dos de copas, carta2 es el rey de bastos y la pinta es copas.

Vencer(carta1,carta2,copas) devuelve cierto; Vencer(carta2,carta1,copas) devuelve falso; Vencer(carta1,carta2,bastos) devuelve falso.

D.N.I. : **CENTRO ASOCIAD**O:

INGENIERIA TECNICA en INFORMATICA de GESTION y SISTEMAS

ASIGNATURA : PROGRAMACION I ORIGINAL

CODIGO CARRERA: **41** = GESTION CONVOCATORIA: FEBRERO 1ª PP

40 = SISTEMAS SEMANA: 2ª Semana TIPO EXAMEN: **2**

CODIGO ASIGNATURA: **103**

DURACIÓN : 2 horas FECHA: 8−II−1999

MATERIAL AUXILIAR: NINGUNO HORA: 11:30 horas

¡ATENCIÓN! PONGA EL TIPO DE EXAMEN EN LA HOJA DE LECTURA OPTICA.

Contéstese el test en la hoja de lectura óptica.

El test es **ELIMINATORIO** (son necesarias 7 respuestas correctas para pasarlo), y aporta el 40% de la nota final.

Sólo hay una respuesta correcta en cada pregunta.

1. Respecto a la sentencia:

Correcto := calculo(matriz) IN grupo;

Se puede decir en cualquier caso que:

A.− **Correcto es de tipo BOOLEAN**

B.− Correcto y grupo son de tipos compatibles

C.− La función calculo devuelve un tipo BOOLEAN

D.− grupo es de tipo enumerado

2. En la sentencia:

Pagina.Imprimir;

A.− **Pagina es un módulo**

B.− Pagina es un registro

C.− Imprimir es un campo variante

D.− Imprimir es una función

3. El invariante de una iteración es la condición que se debe cumplir siempre:

A.− **Antes y después de cada nueva repetición**

B.− Sólo después de cada repetición

C.− Sólo antes de cada repetición

D.− En cualquier punto del bucle iterativo

4. Dado el siguiente fragmento de código:

TYPE INTEGER=[0..10];

VAR K: ARRAY [10..20] OF INTEGER;

K[15]:= 10;

A.− **Es correct**o.

B.− El tipo de datos [10..20] utilizado en la declaración es incorrecto.

C.− El acceso al elemento 15 es incorrecto.

D.− La asignación del valor 10 produce un error de incompatibilidad de tipos.

5. Dado el siguiente fragmento de código:

TYPE INTEGER=('0','1','2','3','4','5');

VAR K: ARRAY [2..4] OF INTEGER;

K[3]:= 2;

A.− **La redefinición del tipo INTEGER es incorrect**a.

B.− La declaración de la variable K es incorrecta.

C.− K[3] accede al tercer elemento del vector K.

D.− Es correcto.

6. Dado el siguiente fragmento de código:

VAR a:INTEGER;

TYPE CHAR=SET OF ['1'..'9'];

VAR b:CHAR;

b:=CHAR{'1'};

a:=VAL(CHAR, '1');

A.− **La asignación del valor a la variable a es incorrect**a.

B.− Es correcto y la variable a toma el valor 0.

C.− Es correcto y la variable a toma el valor 1.

D.− Es correcto y la variable a toma el valor '1'.

7. Dada la siguiente declaración

VAR T:tipodias

Del siguiente fragmento de código

dato=dias{T}

podemos decir que

A.− es una expresión condicional

B.− dato debe ser una constante

C.− es una sentencia de asignación

D.− es una definición de un tipo

8. Dada la siguiente declaración

VAR p1:POINTER TO INTEGER;

p2:POINTER TO REAL;

BEGIN

NEW(p1); NEW(p2);

La sentencia correcta será

A.− p1^:=TRUNC(p2^);

B.− p1:=TRUNC(p2);

C.− p1:=p2;

D.− p2:=FLOAT(p1);

9. Cuál de las siguientes parejas de operadores tienen siempre resultados del

mismo tipo

A.− AND , IN

B.− * , +

C.− *, IN

D.− OR, INCL

10. Dada la siguiente declaración

VAR dato:ARRAY[1..10] OF INTEGER;

Con la siguiente sentencia

FOR cont:=1 TO 10 DO

dato[cont]:=dato[cont+1]

END

A.− Cometemos un error de acceso a los elementos del vector

B.− Trasladamos los elementos del vector una posición a la izquierda

C.− Trasladamos los elementos del vector una posición a la derecha

D.− Manipulamos el índice del vector por referencia

EJERCICIO DE PROGRAMACIÓN

Realizar un **programa completo** en Modula 2 que gestione la asignación de butacas de un recinto en el que hay 5 filas de 7 butacas cada una. El recinto será un **dato**

encapsulado con las siguientes operaciones posibles ante la solicitud de un cliente (1 / 2 / 3):

1. Informe de la ocupación por pantalla. (Por ejemplo: Quedan 7 butacas vacías).

2. Asignación de una butaca libre.

3. Liberación de una butaca ocupada.

RELLENE EN ESTA HOJA Y EN LA HOJA DE LECTURA ÓPTICA LOS SIGUIENTES DATOS:

Apellido**s:.....**Tlfno.**:.....**

Nombre**:.....**D.N.I.**:.....**

Código Carrera: **40 (Sist.)** Código Asignatura: **103** Convocatoria: **Septiembre 1ªPP**

41(Gest.) Semana: **1ª**

Tipo de Examen: **1**

ð El **test** debe ser contestado **en la hoja de lectura óptic**a. Sólo una de las cuatro respuestas posibles de cada pregunta es correcta.

ð El test es eliminatorio y aporta un 40% de la nota final. Son necesarias 7 respuestas correctas para que se corrija el ejercicio.

ð La solución del ejercicio se realizará en el reverso de esta hoja. **No se corregirán hojas auxiliare**s.

ENTREGUE ÚNICAMENTE ESTA HOJA Y LA HOJA DE LECTURA ÓPTICA sin grapar

TEST (cada respuesta correcta: 1punto; respuesta incorrecta o en blanco: 0 puntos)

1. Dado el siguiente fragmento de código:

CASE j OF

enero..junio:INC(j);|

diciembre..julio:DEC(j);

ELSE

END;

A.− la variable j tiene que ser de tipo ordinal para ser correcto

B.− es necesaria alguna sentencia tras ELSE

C.− la sentencia ELSE no se puede usar dentro de una instrucción CASE

D.− los rangos establecidos son incorrectos

2. Para que en Modula−2, la siguiente definición de tipo sea la de una tabla:

TYPE TipVect= ARRAY TipInd OF TipElem

A.− TipInd debe ser cualquier tipo ordinal y TipElem RECORD

B.− TipInd debe ser cualquier tipo predefinido y TipElem RECORD

C.− TipInd debe ser un tipo ordinal definido por el usuario y TipElem de cualquier

tipo

D.− TipInd debe ser RECORD y TipElem de cualquier tipo

3. Dado el siguiente procedimiento:

PROCEDURE proc(a,b:INTEGER);

VAR aux:INTEGER;

BEGIN

aux:=a+b+c;

c:=aux;

END proc;

Para que fuese puro:

A.− la variable c se debería pasar por referencia

B.− Es ya un procedimiento puro

C.− bastaría con pasar las variables a y b por referencia

D.− Todas las variables, incluida aux, deben pasarse por referencia

4. Sabiendo que en Modula−2 existe el tipo predefinido BOOLEAN:

TYPE BOOLEAN=(FALSE, TRUE)

Se puede afirmar:

A.− ORD(FALSE)=0

B.− ORD(FALSE)=1

C.− El tipo BOOLEAN no es un tipo ordinal

D.− Con los tipos predefinidos no se puede utilizar la ORD

5. La compilación segura:

A.− Necesita un modulo de definición

B.− Tiene como objetivo comprobar la compatibilidad de tipos

C.− Produce un programa objeto más eficiente

D.− Mejora la reutilización

6. Dado el siguiente fragmento de código:

TYPE t1=RECORD c1, c2:REAL; END;

t2=POINTER TO REAL;

t3=POINTER TO t1;

VAR a:t1; b:t2; c:t3;

...

NEW(b); NEW(c);

La asignación correcta es:

A.− c^.c1:=a.c2;

B.− b:= c^.c1;

C.− a:=c;

D.− c.c1:=a.c2;

7. Dado la siguiente declaración de variables:

VAR c1:T1; c2:T2;

Después de ejecutar

Leer(c1); Leer(c2);

las variables c1 y c2 toman los valores esto y aquello respectivamente.

A la vista del resultado, el procedimiento Leer tendrá como argumento:

A.− (VAR a: ARRAY OF CHAR)

B.− (a:T1,T2)

C.− (VAR a:T1,T2)

D.− (a: ARRAY OF CHAR)

8. Después de ejecutar el siguiente fragmento de código:

VAR pt1, pt2, pt3: POINTER TO REAL;

...

NEW(pt1); pt1^:=3.0; pt2:=pt1; pt3:=pt2;

DISPOSE(pt2);

A.− Se liberan los punteros pt1, pt2 y pt3

B.− Sólo se libera pt3

C.− Sólo se libera pt3 y pt2

D.− Se produce un error, y no se libera la memoria

9. La definición completa de un tipo opaco que se declara en un módulo de

definición debe completarse en :

A.− El módulo de implementación correspondiente

B.− El módulo principal del programa

C.− El propio módulo de definición

D.− La declaración también se hace en el módulo de implementación

10. Dado el siguiente procedimiento:

PROCEDURE

Calcular (VAR A: INTEGER; B: INTEGER) : INTEGER;

BEGIN

A := B MOD 2; B := A * A + 3 * B − 6; RETURN A+B

END Calcular;

El valor de X tras ejecutar X:=4; X:=Calcular(X,X); será:

A.− 6

B.− 0

C.− 4

D.− 12

EJERCICIO DE PROGRAMACIÓN (10 puntos)

Construir un dato encapsulado que sea una tabla de 50 elementos, en la que se almacenan los datos de una persona: nombre,

apellido1, apellido2, dirección y teléfono. Las operaciones serán añadir un elemento, eliminar un elemento y un procedimiento de

búsqueda selectiva por nombre o por número de teléfono. Si se encuentra la persona en la tabla se devolverá cierto y se mostrará

toda la información disponible de esa persona y si no se devolverá falso.

D.N.I. : **CENTRO ASOCIAD**O:

INGENIERIA TECNICA en INFORMATICA de GESTION y SISTEMAS

ASIGNATURA : PROGRAMACION I ORIGINAL

CODIGO CARRERA: **41** = GESTION CONVOCATORIA: FEBRERO 1ª PP

40 = SISTEMAS SEMANA: 1ª Semana TIPO EXAMEN: **A**

CODIGO ASIGNATURA: **103**

DURACIÓN : 2 horas FECHA: 25−I−1999

MATERIAL AUXILIAR: NINGUNO HORA: 11:30 horas

¡ATENCIÓN! PONGA EL TIPO DE EXAMEN EN LA HOJA DE LECTURA OPTICA.

Contéstese el test en la hoja de lectura óptica.

El test es **ELIMINATORIO** (son necesarias 7 respuestas correctas para pasarlo), y aporta el 40% de la nota final.

Sólo hay una respuesta correcta en cada pregunta.

NOTA: La solución del ejercicio se realizará en el reverso de esta hoja. NO se corregirá lo que exceda de

1. Respecto a la sentencia:

Correcto := calculo(matriz) IN grupo;

Se puede decir en cualquier caso que:

A.− matriz y grupo son de tipos compatibles

B.− calculo es un procedimiento que devuelve como resultado matriz

C.− La función calculo devuelve un valor enumerado

D.− grupo es de tipo BOOLEAN

2. Los metasimbolos son:

A.− Elementos de la notación BNF

B.− Elementos de la programación lógica

C.− Elementos de la programación funcional

D.− Parte del modelo de flujo de datos

3. La reutilización se consigue mediante desarrollo:

A.− Sólo ascendente

B.− Ascendente o descendente

C.− Sólo descendente

D.− Robusto

4. Los tipos opacos:

A.− Son de acceso secuencial

B.− Son datos persistentes

C.− Son punteros

D.− Son tipos abstractos de datos

5. Dado el siguiente fragmento de código:

```
FOR i:=0 TO 1 BY 0.1 DO
WriteString("Esto se escribe 10 veces");
END
```

A.− Se produce un error por incompatibilidad de tipos.

B.− El cuerpo del bucle se ejecuta 10 veces.

C.− La frase se escribe una vez, pero al incrementar el índice se produce un error.

D.− Aunque no hay errores, el cuerpo del bucle nunca se ejecuta.

6. Dado el siguiente fragmento de código:

```
VAR a:CHAR;
TYPE CHAR=SET OF ['1'..'9'];
VAR b:CHAR;

b:=CHAR{'3'..'5'};
a:=b;
```

A.− Se produce un error en la asignación a:=b.

B.− La declaración de las variables es incompatible.

C.− Se produce un error en la asignación b:=CHAR{'3'..'5'}.

D.− Es correcto.

7. A la vista únicamente de la siguiente declaración:

```
VAR K:POINTER TO SET OF [1..10];
```

Sólo se puede decir que:

A.− Es incorrecta.

B.− K es de tipo anónimo.

C.− K es un dato encapsulado.

D.− K es de tipo opaco.

8. Dada la siguiente fragmento de código

dato=dias{L};

podemos decir que

A.− dato no tiene tipo

B.− L puede ser una variable o una constante

C.− dias es el nombre del tipo referencial

D.− dias{L} es un valor constante de tipo conjunto

9. Dado el siguiente módulo

DEFINITION MODULE Segundo;

PROCEDURE Previo(VAR dato:tipodato):

END Segundo.

A.− Es correcto, tipodato es un tipo opaco

B.− Es erróneo, se necesita la declaración de tipodato

C.− Es erróneo tipodato es un tipo anónimo

D.− Es erróneo dato debería ser por valor

10. La cabecera del subprograma WriteString para imprimir cadenas de

caracteres podría ser

A.− PROCEDURE WriteString(a:POINTER TO CHAR);

B.− PROCEDURE WriteString(VAR a:ARRAY[1..1000]OF CHAR);

C.− PROCEDURE WriteString(a:CHAR);

D.− PROCEDURE WriteString(a:ARRAY OF CHAR);

EJERCICIO DE PROGRAMACIÓN

En el módulo Juegos, se dispone del tipo abstracto de datos CartaBaraja, que representa una carta de la baraja española. También se dispone de

dos operaciones asociadas: PonerPalo y PonerTriunfo.

PonerPalo, establece el palo de la carta: oros, copas, espadas o bastos.

Por ejemplo: PonerPalo(carta,oros).

PonerTriunfo, establece el triunfo de la carta: as, dos,, sota, caballo o rey.

Por ejemplo: PonerTriunfo(carta,rey)

Se pide crear en el módulo principal una baraja de 40 cartas y la operación Vencer. Una carta vence a otra cuando su triunfo es mayor, excepto

cuando una de las cartas es de la pinta que entonces gana aún cuando su triunfo sea menor. Si dos cartas tienen igual triunfo vence cualquiera

de las dos.

Ejemplo: carta1 es el dos de copas, carta2 es el rey de bastos y la pinta es copas.

Vencer(carta1,carta2,copas) devuelve cierto; Vencer(carta2,carta1,copas) devuelve falso; Vencer(carta1,carta2,bastos) devuelve falso.

RELLENE EN ESTA HOJA Y EN LA HOJA DE LECTURA ÓPTICA LOS SIGUIENTES DATOS:

Apellido**s:.....**Tlfno.**:.....**

Nombre**:.....**D.N.I.**:.....**

Código Carrera: **40 (Sist.)** Código Asignatura: **103** Convocatoria: **Septiembre 1ªPP**

41(Gest.) Semana: **1ª**

Tipo de Examen: **A**

ð El **test** debe ser contestado **en la hoja de lectura óptic**a. Sólo una de las cuatro respuestas posibles de cada pregunta es correcta.

ð El test es eliminatorio y aporta un 40% de la nota final. Son necesarias 7 respuestas correctas para que se corrija el ejercicio.

ð La solución del ejercicio se realizará en el reverso de esta hoja. **No se corregirán hojas auxiliare**s.

ENTREGUE ÚNICAMENTE ESTA HOJA Y LA HOJA DE LECTURA ÓPTICA sin grapar

TEST (cada respuesta correcta: 1punto; respuesta incorrecta o en blanco: 0 puntos)

1. Dado el siguiente fragmento de código:

```
CASE j OF
enero..junio:INC(j);|
diciembre..julio:DEC(j);
ELSE
END;
```

A.− la variable j tiene que ser de tipo ordinal para ser correcto

B.− es necesaria alguna sentencia tras ELSE

C.− la sentencia ELSE no se puede usar dentro de una instrucción CASE

D.− los rangos establecidos son incorrectos

2. Para que en Modula−2, la siguiente definición de tipo sea la de una tabla:

```
TYPE TipVect= ARRAY TipInd OF TipElem
```

A.− TipInd debe ser un tipo ordinal definido por el usuario y TipElem de cualquier tipo

B.− TipInd debe ser cualquier tipo predefinido y TipElem RECORD

C.− TipInd debe ser cualquier tipo ordinal y TipElem RECORD

D.− TipInd debe ser RECORD y TipElem de cualquier tipo

3. Dado el siguiente procedimiento:

```
PROCEDURE proc(a,b:INTEGER);
VAR aux:INTEGER;
BEGIN
aux:=a+b+c;
c:=aux;
END proc;
```

Para que fuese puro:

A.− Todas las variables, incluida aux, deben pasarse por referencia

B.− Es ya un procedimiento puro

C.− bastaría con pasar las variables a y b por referencia

D.− la variable c se debería pasar por referencia

4. Sabiendo que en Modula−2 existe el tipo predefinido BOOLEAN:

TYPE BOOLEAN=(FALSE, TRUE)

Se puede afirmar:

A.− ORD(FALSE)=0

B.− ORD(FALSE)=1

C.− El tipo BOOLEAN no es un tipo ordinal

D.− Con los tipos predefinidos no se puede utilizar la ORD

5. La compilación segura:

A.− Mejora la reutilización

B.− Produce un programa objeto más eficiente

C.− Tiene como objetivo comprobar la compatibilidad de tipos

D.− Necesita un modulo de definición

6. Dado el siguiente fragmento de código:

TYPE t1=RECORD c1, c2:REAL; END;

t2=POINTER TO REAL;

t3=POINTER TO t1;

VAR a:t1; b:t2; c:t3;

...

NEW(b); NEW(c);

La asignación correcta es:

A.− c^.c1:=a.c2;

B.− b:= c^.c1;

C.− a:=c;

D.− c.c1:=a.c2;

7. Dado la siguiente declaración de variables:

VAR c1:T1; c2:T2;

Después de ejecutar

Leer(c1); Leer(c2);

las variables c1 y c2 toman los valores esto y aquello respectivamente.

A la vista del resultado, el procedimiento Leer tendrá como argumento:

A.− (VAR a: ARRAY OF CHAR)

B.− (a:T1,T2)

C.− (VAR a:T1,T2)

D.− (a: ARRAY OF CHAR)

8. Después de ejecutar el siguiente fragmento de código:

VAR pt1, pt2, pt3: POINTER TO REAL;

...

NEW(pt1); pt1^:=3.0; pt2:=pt1; pt3:=pt2;

DISPOSE(pt2);

A.− Sólo se libera pt3

B.− Se liberan los punteros pt1, pt2 y pt3

C.− Sólo se libera pt3 y pt2

D.− Se produce un error, y no se libera la memoria

9. La definición completa de un tipo opaco que se declara en un módulo de

definición debe completarse en :

A.− La declaración también se hace en el módulo de implementación

B.− El módulo principal del programa

C.− El propio módulo de definición

D.− El módulo de implementación correspondiente

10. Dado el siguiente procedimiento:

PROCEDURE

Calcular (VAR A: INTEGER; B: INTEGER) : INTEGER;

BEGIN

A := B MOD 2; B := A * A + 3 * B − 6; RETURN A+B

END Calcular;

El valor de X tras ejecutar X:=4; X:=Calcular(X,X); será:

A.− 4

B.− 0

C.− 6

D.− 12

EJERCICIO DE PROGRAMACIÓN (10 puntos)

Construir un dato encapsulado que sea una tabla de 50 elementos, en la que se almacenan los datos de una persona: nombre,

apellido1, apellido2, dirección y teléfono. Las operaciones serán añadir un elemento, eliminar un elemento y un procedimiento de

búsqueda selectiva por nombre o por número de teléfono. Si se encuentra la persona en la tabla se devolverá cierto y se mostrará

toda la información disponible de esa persona y si no se devolverá falso.

INGENIERIA TECNICA en INFORMATICA de GESTION y SISTEMAS

ASIGNATURA : PROGRAMACION I ORIGINAL

CODIGO CARRERA: **41** = GESTION CONVOCATORIA: FEBRERO 1ª PP

40 = SISTEMAS SEMANA: 2ª Semana TIPO EXAMEN: **E**

CODIGO ASIGNATURA: **103**

DURACIÓN : 2 horas FECHA: 8−II−1999

MATERIAL AUXILIAR: NINGUNO HORA: 11:30 horas

¡ATENCIÓN! PONGA EL TIPO DE EXAMEN EN LA HOJA DE LECTURA OPTICA.

Contéstese el test en la hoja de lectura óptica.

El test es **ELIMINATORIO** (son necesarias 7 respuestas correctas para pasarlo), y aporta el 40% de la nota final.

Sólo hay una respuesta correcta en cada pregunta.

NOTA: La solución del ejercicio se realizará en el reverso de esta hoja. NO se corregirá lo que exceda de

1. Respecto a la sentencia:

Correcto := calculo(matriz) IN grupo;

Se puede decir en cualquier caso que:

A.− Correcto es de tipo BOOLEAN

B.− Correcto y grupo son de tipos compatibles

C.− La función calculo devuelve un tipo BOOLEAN

D.− grupo es de tipo enumerado

2. En la sentencia:

Pagina.Imprimir;

A.− Imprimir es un campo variante

B.− Pagina es un registro

C.− Pagina es un módulo

D.− Imprimir es una función

3. El invariante de una iteración es la condición que se debe cumplir siempre:

A.− Antes y después de cada nueva repetición

B.− Sólo después de cada repetición

C.− Sólo antes de cada repetición

D.− En cualquier punto del bucle iterativo

4. Dado el siguiente fragmento de código:

```
TYPE INTEGER=[0..10];

VAR K: ARRAY [10..20] OF INTEGER;
```

K[15]:= 10;

A.− El acceso al elemento 15 es incorrecto.

B.− El tipo de datos [10..20] utilizado en la declaración es incorrecto.

C.− Es correcto.

D.− La asignación del valor 10 produce un error de incompatibilidad de tipos.

5. Dado el siguiente fragmento de código:

TYPE INTEGER=('0','1','2','3','4','5');

VAR K: ARRAY [2..4] OF INTEGER;

K[3]:= 2;

A.− La redefinición del tipo INTEGER es incorrecta.

B.− La declaración de la variable K es incorrecta.

C.− K[3] accede al tercer elemento del vector K.

D.− Es correcto.

6. Dado el siguiente fragmento de código:

VAR a:INTEGER;

TYPE CHAR=SET OF ['1'..'9'];

VAR b:CHAR;

b:=CHAR{'1'};

a:=VAL(CHAR, '1');

A.− Es correcto y la variable a toma el valor 1.

B.− Es correcto y la variable a toma el valor 0.

C.− La asignación del valor a la variable a es incorrecta.

D.− Es correcto y la variable a toma el valor '1'.

7. Dada la siguiente declaración

VAR T:tipodias

Del siguiente fragmento de código

dato=dias{T}

podemos decir que

A.− dato debe ser una constante

B.− es una expresión condicional

C.− es una sentencia de asignación

D.− es una definición de un tipo

8. Dada la siguiente declaración

VAR p1:POINTER TO INTEGER;

p2:POINTER TO REAL;

BEGIN

NEW(p1); NEW(p2);

La sentencia correcta será

A.− p1:=TRUNC(p2);

B.− p1^:=TRUNC(p2^);

C.− p1:=p2;

D.− p2:=FLOAT(p1);

9. Cuál de las siguientes parejas de operadores tienen siempre resultados del

mismo tipo

A.− OR, INCL

B.− * , +

C.− *, IN

D.− AND , IN

10. Dada la siguiente declaración

VAR dato:ARRAY[1..10] OF INTEGER;

Con la siguiente sentencia

FOR cont:=1 TO 10 DO

dato[cont]:=dato[cont+1]

END

A.− Manipulamos el índice del vector por referencia

B.− Trasladamos los elementos del vector una posición a la izquierda

C.− Trasladamos los elementos del vector una posición a la derecha

D.− Cometemos un error de acceso a los elementos del vector

EJERCICIO DE PROGRAMACIÓN

Realizar un **programa completo** en Modula 2 que gestione la asignación de butacas de un recinto en el que hay 5 filas de 7 butacas cada una. El recinto será un **dato**

encapsulado con las siguientes operaciones posibles ante la solicitud de un cliente (1 / 2 / 3):

1. Informe de la ocupación por pantalla. (Por ejemplo: Quedan 7 butacas vacías).

2. Asignación de una butaca libre.

3. Liberación de una butaca ocupada.

Enunciado

En el módulo Juegos, se dispone del tipo abstracto de datos

CartaBaraja, que representa una carta de la baraja española.

También se dispone de dos operaciones asociadas: PonerPalo y

PonerTriunfo.PonerPalo, establece el palo de la carta: oros, copas,

espadas o bastos.Por ejemplo: PonerPalo(carta,oros).PonerTriunfo,

establece el triunfo de la carta: as, dos,, sota, caballo o rey.Por

ejemplo: PonerTriunfo(carta,rey).Se pide crear en el módulo

principal una baraja de 40 cartas y la operación Vencer. Una carta

vence a otra cuando su triunfo es mayor, excepto cuando una de las

cartas es de la "pinta" que entonces gana aún cuando su triunfo sea

menor. Si dos cartas tienen igual triunfo vence cualquiera de las

dos.Ejemplo: carta1 es el dos de copas, carta2 es el rey de bastos y la

"pinta" es copas.Vencer(carta1,carta2,copas) devuelve cierto;

Vencer(carta2,carta1,copas) devuelve falso;

Vencer(carta1,carta2,bastos) devuelve falso.

Solución

```
MODULE Examen;
FROM Juegos IMPORT CartaBaraja, PonerPalo, PonerTriunfo,
TipoPalo, TipoTriunfo, PedirPalo, PedirTriunfo;
TYPE
Baraja=ARRAY[1..40] OF CartaBaraja;
VAR
B1:Baraja;
C1:TipoPalo;
C2:TipoTriunfo;
PROCEDURE
Vencer(Carta1,Carta2:CartaBaraja;Pinta:TipoPalo);
BEGIN
IF (PedirPalo(Carta1)=PedirPalo(Carta2)) THEN
IF (PedirTriunfo(Carta1)> PedirTriunfo(Carta2)) THEN
RETURN TRUE;
ELSE
RETURN FALSE;
END;
ELSE
IF (PedirPalo(Carta1)=Pinta) THEN
```

```
RETURN TRUE;
ELSIF (PedirPalo(Carta2)=Pinta ) THEN
RETURN FALSE;
ELSE
RETURN TRUE;
END;
END;
END Vencer;
BEGIN
(* Rellenado de las cuarenta cartas de la baraja *)
FOR C1:= oros TO bastos DO
FOR C2:= as TO rey DO
PonerPalo (B1[ORD(C2)+1+ORD(C1)*10],C1);
PonerTriunfo (B1[ORD(C2)+1+ORD(C1)*10],C2);
END;
END;
```

15