Fundamentos de la programación y la informática
Examen Parcial. 17 de Noviembre de 2022
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

- Ejecuta en un terminal la orden
  `~mortuno/prepara`
- Comprueba que esto ha dejado en tu cuenta los ficheros
  `~/fpi.nov.22/logica.TULOGIN.txt`, donde deberás resolver el ejercicio 1 [1]..
  `~/fpi.nov.22/pegatina.TULOGIN.txt`, donde deberás resolver el ejercicio 2.

Ejercicio 1 (2 puntos)

Sean las expresiones

```
e1 := not (a and not b) and not (not c or d) and not (not e and f) ;

e2 := not (x >  4) and not (y <= 20);
```

1. A partir de e1, aplica De Morgan y escribe una expresión lógica equivalente, intentando que sea más clara, de nombre s1
2. A partir de e2, escribe una expresión lógica equivalente, más legible, de nombre s2
3. Escribe una expresión n1 que sea la negación de s1
4. Escribe una expresión n2 que sea la negación de s2

Solución

```
s1: (not a or b) and (c and not d) and (e or not f)

s2: (x <= 4) and (y > 20)

n1: (a and not b) or ( not c or d) or (not e and f)

n2: (x > 4) or (y <= 20)
```

Ejercicio 2 (8 puntos)

Como seguramente sabes, los coches en España llevan una pegatina que proporciona la DGT, para clasificarlos según su nivel de emisiones. Las categorías son *cero*, *eco*, *a*, *b* y *c*. Se asignan según la siguiente tabla:

- cero

  Vehículos eléctricos.

- eco

  Vehículos híbridos o que consumen gas.

[1] TULOGIN será tu nombre de usuario en el laboratorio

- c

  Vehículos de gasolina, matriculados en año 2006, incluido, o posterior.

  Vehículos diésel, matriculados en año 2014, incluido, o posterior.

- b

  Vehículos de gasolina, matriculados desde el año 2000, incluido, hasta el año 2006, excluido.

  Vehículos diésel, matriculados desde el año 2006, incluido, hasta el 2014, excluido.

- a

  Vehículos que no cumplen ninguno de los casos anteriores

Escribe un programa en pascal en el fichero `~/fpi.nov.22/pegatina.TULOGIN.txt`, según la siguiente especificación.

1. El cuerpo principal del programa tendrá una constante de tipo cadena llamada *Motor* que podrá contener alguno de los siguientes valores: *electrico*, *hibrido*, *gas*, *gasolina*, *diesel*

   (Exactamente así, en español, en minúscula y sin tildes)

2. El cuerpo principal también definirá una constante de tipo entero llamada *Matricula*, que contendrá el año de matriculación.

3. El programa tendrá un subprograma que comprobará la precondición de que *Motor* contenga un valor correcto, conforme con lo requerido en el punto 1. La forma en la que este subprograma indica el cumplimiento/incumplimiento de la precondición, queda a tu elección.

4. El programa tendrá un subprograma que comprobará la precondición de que *Matricula* contenga un número entero entre 1900 y 2030, ambos inclusive.

5. El programa tendrá un subprograma que recibirá un parámetro *motor* y otro *matricula*, y devolverá

   - La cadena *cero*, *eco*, *a*, *b* o *c*, según la tabla indicada.
   - Un código entero de éxito/error: 0 si todo ha ido bien, otro valor si ha habido algún problema.

6. El programa tendrá un subprograma que llamará al subprograma descrito en el apartado 5 y mostrará en pantalla todo lo que creas que debería mostrar.

Observaciones

- El programa no pide ningún valor por teclado al usuario.
- El programa tiene que tener al menos los subprogramas indicados. Si crees que debería tener alguno más y/o que estos subprogramas deberían llamar a otros subprogramas, adelante.
- El código fuente del programa y los comentarios pueden estar en español, o si te animas, mejor en inglés. Pero las constantes tienen que estar en español, como indica la especificación.

Solución

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}


program pegatina;
const
    MatricMax:integer=2030;   // Máximo año de matriculación correcto
    MatricMin:integer=1900;   // Mínimo año de matriculación correcto

function precond_motor(tipomot:string):boolean;
begin
    result := (tipomot='hibrido') or (tipomot='electrico') or (tipomot='gas')
        or (tipomot='gasolina') or (tipomot='diesel');
end;

function precond_matricula(matricula:integer):boolean;
begin
    result := (matricula>=MatricMin) and (matricula<=MatricMax);
end;

function clasifica_gasolina(matricula:integer):string;
begin
    if (matricula>=2006) then // Valores fijados en la ley
        result:='c';

    if (matricula>=2000) and (matricula<2006) then
        result:='b';

    if matricula < 2000 then
        result := 'a';
end;

function clasifica_diesel(matricula:integer):string;
begin
    if (matricula>=2014) then  // Valores fijados en la ley
        result:='c';

    if (matricula>=2006) and (matricula<2014) then
        result:='b';

    if matricula < 2006 then
        result:='a';
end;

function asigna_pegatina(motor:string;matricula:integer):string;
begin
    if motor = 'electrico' then
        result := 'cero';

    if (motor = 'gas') or (motor = 'hibrido') then
        result := 'eco';

    if motor = 'gasolina' then
        result := clasifica_gasolina(matricula);
```

```
    if motor = 'diesel' then
        result := clasifica_diesel(matricula);
end;

procedure clasifica_coche(motor:string; matricula:integer;
var pegat:string; var codigo:integer);
begin
    if not precond_motor(motor) then
        codigo := 1;  // Error, motor incorrecto

    if not precond_matricula(matricula) then
        codigo := 2;  // Error, año matriculación incorrecto

    if (precond_motor(motor)) and (precond_matricula(matricula))
    then begin
        codigo := 0;  // ok
        pegat := asigna_pegatina(motor, matricula);
    end;

end;

procedure escribe_resultados(motor: string; matricula:integer;
    pegat: string; codigo: integer);
begin
   writeln('Motor: ',motor);
   writeln( 'Año de matriculación: ',matricula);

   if codigo = 0 then
       writeln('Pegatina: ', pegat);

   if codigo = 1 then
       writeln('Error: motor no reconocido');

   if codigo = 2 then
       writeln('Error: año de matriculación incorrecto');
end;
       // Programa principal
const
    Motor = 'diesel';
    Matricula = 2007 ;
var
    codigo: integer;  // éxito / error
    pegat: string;

begin
    clasifica_coche(Motor, Matricula, pegat, codigo);
    escribe_resultados(Motor, Matricula, pegat, codigo);
end.
```

Criterio general de corrección

El criterio aplicado en la corrección es el siguiente:

- Es imprescindible que el programa, al menos, compile.
- Los diseños que no se corresponden con lo requerido en el enunciado, y/o defectuosos, tendrán una penalización en la nota pero no necesariamente el suspenso. Un diseño pobre pero que más o menos *funciona* en la mayoría de los casos, puede tener una nota superior a 5.
- Un error que imposibilita por completo dar una solución correcta, tendrá una nota inferior a 5.

Observaciones. Errores frecuentes

A continuación describiremos algunos problemas habituales en los ejercicios presentados.

1. En la solución propuesta, siguiendo una pauta de diseño muy recomendable, el cuerpo del programa principal no calcula nada, no tiene ningún *if then else*. Se limita a llamar a otros procedimientos. Solamente en ejemplos muy muy sencillos encontraremos código que toma decisiones en el cuerpo principal.

   Metáfora: el consejero delegado de una empresa no hace fotocopias. Excepto tal vez que sea un autónomo *micro empresa*.

2. Los extremos de los rangos de años de matriculación (2000, 2006, 2014), normalmente los definiríamos como constantes, para evitar el uso de *numeros mágicos*. Pero como estos valores están fijado *a fuego* en la ley, en ningún caso cambiarían y está muy claro cuál es su origen, en este caso hemos preferido dejarlos como constante literales numéricas (*tal cual*). También sería razonable defender el criterio estricto y usar constantes.

   Lo que no tendría mucho sentido es usar constantes para las cadenas, como

   ```
   const
       M1 = 'electico';
       M2 = 'hibrido';
       M3 = 'gas';
       (etc)
   ```

3. Los principiantes suelen escribir las precondiciones con una sentencia *if then else* como esta:

   ```
   if  (matricula>=MatricMin) and (matricula<=MatricMax) then
       result := TRUE
   else
       result := FALSE;
   ```

   No es un error, no penaliza. Pero es innecesario.

4. El principiante también tiende al siguente diseño muy pobre: cuando detecta un el fallo de una precondición, lo escribe en pantalla y el programa concluye con `halt`

   ```
   if  (matricula>=MatricMin) and (matricula<=MatricMax) then
       result := TRUE
   else begin
       writeln('Error: año de matriculación fuera de rango');
       halt;
   end;
   ```

Observa que en la solución propuesta, cuando una precondición falla simplemente se asigna un valor a un parámetro de salida, o al valor devuelto por la función, pero la ejecución sigue.

5. El siguiente error es mucho más severo y sí puede invalidar por completo el ejercicio: cuando el programa detecta un error, lo escribe en pantalla. Pero no hace nada más. Con lo que el programa se sigue ejecutando, como si el valor fuera correcto. Ejemplo:

```
if  (matricula>=MatricMin) and (matricula<=MatricMax) then
    ok := TRUE
else
    writeln('Error: año de matriculación fuera de rango');
    // ¡¡MAL!!  Falta darle a ok el valor FALSE
```

6. Anidar sentencias *if then else* dificulta la lectura del código y por tanto aumenta el riesgo de cometer errores lógico. En esta solución lo hemos evitado por completo:

   - Las precondiciones son funciones aparte. Esto casi siempre es conveniente. Primero comprobamos que los valores son correctos, y luego, en un subprograma diferente, hacemos el cálculo que corresponda. Sin mezclar ambas cosas.
   - Hemos escrito una función *clasifica_diesel* y otra *clasifica_gasolina*. Una división diferente pero también razonable sería escribir una función para las *pegatinas* de cierta complejidad. P.e.

     ```
     function pegatina_c(motor:string, matricula:integer):boolean;
     begin
         result :=
         (  (motor = gasolina) and (matricula >= 2006)  )
         or
         ( (motor = diesel) and (matricula >= 2014) )
     end;
     ```

7. El siguente error también es relativamente frecuente.

```
procedure tipo_motor(Motor:string; var codigo:integer);
begin
    if Motor = 'electrico' then
        codigo := 0;
    if Motor = 'hibrido' then
        codigo := 0;
    if Motor = 'gas' then
        codigo := 0;
    if Motor = 'gasolina' then
        codigo := 0;
    if Motor = 'diesel' then
        codigo := 0
    else begin
        codigo := 1;  // ¡¡MAL!!
    end
end;
```

Un mismo subprograma se puede escribir de muchas formas distintas todas ellas adecuadas. Las sentencias *if then else* pueden ser independientes. O pueden encadenarse sentencias *else if*. Pero mezcladas como en este ejemplo son un error muy severo que invalida por completo el ejercicio.

El programador pensaba que el último *else* se ejecuta cuando todas las condiciones anteriores fallan. Pero no. Se trata de sentencias *if then else* independientes. Este programa devolverá el código 0 cuando el motor sea diésel y el código 1 cuando no sea diésel, destruyendo los valores que se hubieran podido asignar al código en alguno de las cuatro setencias *if then* precedentes.

8. Observa que las funciones *clasifica_gasolina*, *clasifica_diesel* y *asigna_pegatina* no escriben nada en pantalla: se limitan a devolver una cadena.

   Y el procedimiento *clasifica_coche* tampoco escribe nada en pantalla. Aunque sea un procedimiento. Es un *subprograma que calcula*, no debería tener efectos laterales. Lo hemos escrito como procedimiento para que pueda devolver dos valores: la pegatina y el código éxito/error.

   Es un error de diseño severo propio de principiante pensar lo siguiente: *como el código está en un procedimiento, ya puedo escribir en pantalla sin problema.* ¡No!

```
procedure clasifica_gasolina(matricula:integer);
begin
    if (matricula>=2006) then
        writeln('Pegatina c');   // MAL

    if (matricula>=2000) and (matricula<2006) then
        writeln('Pegatina b');   //  MAL

    if matricula < 2000 then
        writeln('Pegatina a');   // MAL
end;
```

   Este es un ejemplo claro de *subprograma que calcula*. No debería escribir, aunque lo hayamos *disfrazado* de procedimiento.

9. El siguiente error es una variante del anterior

```
function clasifica_gasolina(matricula:integer):string;
begin
    if (matricula>=2006) then
        result := ('La pegatina del coche es c');   // MAL

    if (matricula>=2000) and (matricula<2006) then
        result:=('La pegatina del coche es b');   //  MAL

    if matricula < 2000 then
        result:=('La pegatina del coche es a');   // MAL
end;
```

   Una función *calcula* algo, y en este caso, la información a calcular es *a*, *b* o *c*. El texto explicativo para el usuario debería estar en el procedimiento que presenta la información, no en el que la calcula.

10. El siguiente ejemplo contiene, entre otros, un tipo de error muy serio presente en unos cuantos ejercicios

```
procedure mostrar_resultados(var a: string; var b: integer);
begin
        a := 'gasolina';   // ¡¡MAL!!
```

```
        b := 2006;          // ¡¡MAL!!

        if (pre_matricula(b)) and (pre_motor(a)) then begin
                write('El tipo de pegatina es: ');
                writeln(tipo_coche(a,b));
        end
        else
                writeln('Error');
end;
```

El autor demuestra que aún no ha comprendido el uso de los parámetros: si un subprograma ignora los parámetros recibidos y los *machaca* con un valor constante, no tiene ningún sentido.