- 4. Calcule el tiempo de ejecución del programa del punto 3. Mostrar los valores intermedios para llegar a resultado y justificar.
- 5. Indique Verdadero o Falso. Justifique en todos los casos:
  - a. Incluir módulos dentro de un programa implica que el programa es más eficiente que otro programa que realiza la misma tarea pero sin utilizar módulos.
- b. El siguiente programa es válido. program ejercicio; function auxiliar(val:integer): integer; val:= val \* val; auxiliar:= val; procedure calculo(c: Integer; var b:integer); begin b:= b + c DIV 4; end; var a,b:integer; begin a:= 16; b:= 6; calculo(auxiliar(a),b);
- c. No siempre es posible declarar un tipo subrango donde su tipo base sea cualquiera de los tipos simpl
- d. Un programa que utiliza un repeat until puede reescribirse utilizando un while.
- La comunicación entre el programa y los módulos no sólo se puede hacer utilizando parámetros.
- 2) Ambos programas son válidos. El programa A primero declara un Array en el apartado type (variables definidas por el usuario), para luego referenciarlo con una variable "d" en el programa principal. De la misma manera, el programa B declara aquel Array directamente en el apartado var del P. Principal, siendo algo válido en Pascal. En el programa principal, el funcionamiento es el mismo. Pero si queremos modularizar, el programa B no facilita la reutilización de ese arreglo, ya que se encuentra declarado debajo solamente para el P. Principal. En el caso de que el apartado var se encuentre por encima de los módulos, estos podrán usar el arreglo, pero el P. Principal se verá obligado a declarar todas sus variables allí arriba de manera global, siendo algo muy peligroso para la seguridad del programa
- 4) Línea 2: 1UT N=100; C=3; Cuerpo=4 --> 3(100+1) + 100(4) = 703UT --> (3\*100 + 2) + 100(2) = 502UT TOTAL: 1 + 703 + 502 = 1206UT
- a. Verdadero. Modularizar no implica eficiencia respecto a tiempo de ejecución, pero si a memoria. Cuando se modulariza y se declaran variables internas en los módulos, se procura que las variables locales que se alojen en la memoria dinámica solamente durante la ejecución del módulo. En otras palabras, aunque no sea demasiado más eficiente, si lo es si se modulariza y se asignan adecuadamente las variables locales y

- b. Verdadero. La funcion auxiliar calcula el cuadrado de un valor. En pascal se puede pasar una función como parámetro. Todo funciona según lo previsto.
- c. Verdadero. Los únicos subrangos que se pueden declarar en pascal son de tipo integer o de tipo char
- d. Verdadero. Se puede reescribir modificando la comparación que se hace en el repeat until. Por ejemplo.

```
i : integer;
begin i:=0;
      writeln("Hola");
      i := i + 1:
    until (i = 5);
  ..se puede reescribir cómo:
   i : integer;
i:=0;

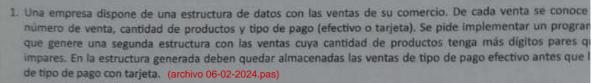
while(i <= 5) do begin

writeln("Hola");

i := i + 1;
```

e. Verdadero. También se pueden utilizar variables globales

Apellido y Nombre ..



 Dados los siguientes programas indique para cada uno si son válidos o no. Además, analice si considera que funcionamiento en ambos programas es el mismo o no. JUSTIFIQUE

```
B
Program uno;
                                     Program dos;
                                      Var
   datos = array [1..100] of
                                       d: array [1..100] of integer;
integer;
 Var
  d: datos;
                                       // Operaciones e invocaciones a
                                     módulos
Begin
                                         para cargar y recorrer el
  //Operaciones e invocaciones a
                                     vector d
módulos
                                      Ende
   para cargar y recorrer el
vector d
End
```

 Calcule e indique la cantidad de memoria estática y dinámica que utiliza el siguiente programa. Mostrar lo valores intermedios para llegar al resultado y justificar.

```
program ejercicio3;
                                                             Char
                                                                       1 byte
  type info = record
                                                             Integer
                                                                       6 bytes
              nombre: string; 255
                                                             Real
                                                                       10 bytes
              nota: integer;
              datos: ^integer; U _ (
                                                             Boolean
                                                                       1 byte
            end:
                                                            String
                                                                       Longitud + 1 byte
       vector = array [1..100] of info; 25,500
                                                            Puntero
                                                                       4 bytes
       v: vector; i,j: integer; e: info;
 var
                                                            Estática:
                                                                        Dinámica:
                    66
                                                                        Suponiendo que i = 100
                                                            e = 265b
 read(e.nombre);
                                                                        (valor máximo que puede
alcanzar i), serían
ocupados 6*100 bytes =
                                                            v = 26500b
 11:=0;
 while (i < 100) and (e.nombre <> 'ZZZ') do
                                                            i = 6
                                                            j = 6
    begin
      read(e.nota);
      e.datos:= nil;
                                                            26777 bytes
     1:= 1 + 1;
     v[1]:= e;
     read(e.nombre);
 3
 " end;
ofor j:= 1 to i do begin
new(v[j].datos);
n v[j].datos^:= v[j].nota MOD 10;
end:
end.
```