Nº Parcial:
Nombre:
CI:

Primer Parcial. Programación 1

Instituto de Computación
Curso 2019

Observaciones:

- El parcial consta de 16 preguntas y **un** ejercicio de programación.
- Cada pregunta tiene una única opción correcta.
- Una respuesta correcta suma 2 puntos. Una respuesta incorrecta resta 0.5 puntos. Una pregunta sin respuesta no afecta el puntaje.
- El ejercicio de programación tiene un total de 8 puntos.
- Debe entregar la planilla con las respuestas junto con la letra y la resolución del ejercicio. Todo debe estar identificado con nombre y cédula.
- En todos los ejercicios se utiliza el lenguaje Pascal tal como fue dado en el curso (ej. circuito corto, `case` con `else`, etc.).

Ejercicio 1 Indique el valor de la siguiente expresión:

```
324 div 10 mod 10 * 324 mod 10 div 10
```

a) 0
b) 2
c) 3
d) 324
e) Da error por dividir entre cero

Ejercicio 2 Considere la siguiente declaración de variables:

```
const N = 10;
var info : array [1 .. N] of char;
    s  : boolean;
    a,i  : integer;
```

Dado el siguiente fragmento de programa:

```
a := 0;
for i := 1 to N do
   case info [i] of
      '#'              : s := not s;
      '0','3','6','9' : if s then a := a + 2;
      '1','4','7'     : a := a + 1
   else s := s or (a > 10)
   end;
writeln (a)
```

Indique cuál afirmación es correcta sobre la ejecución del fragmento, sin importar el valor de `s`.

a) Da error en tiempo de ejecución
b) Si `info` = | '#' | '1' | '3' | '#' | '1' | '3' | '#' | '1' | '3' | '#' |, escribe 4
c) Si `info` = | '#' | '1' | '3' | '#' | '1' | '3' | '#' | '1' | '3' | '#' |, escribe 6
d) Si `info` = | '#' | '1' | '3' | '#' | '1' | '3' | '#' | '#' | '#' | '#' |, escribe 4
e) Si `info` = | '#' | '1' | '3' | '#' | '1' | '3' | '#' | '#' | '#' | '#' |, escribe 6

Ejercicio 3 Dadas las siguientes declaraciones de constantes y variables.

```
const A = 5;
      B = 3.9;
var   w,z : real;
      i,j : integer;
```

Indique en cuál de las siguientes asignaciones **no** ocurre ninguna coerción implícita de enteros a reales.

a) `w := A / B`
b) `z := z - A`
c) `i := A + trunc(B)`
d) `z := j div A * i`
e) `j := i + trunc(j)`

Ejercicio 4 Dadas las siguientes declaraciones de variables.

```
var x    : Real;
    y, i : Integer;
    car  : Char;
    bb   : Boolean;
```

Indique cuál asignación a `bb` es válida. Asuma que todas las variables han sido inicializadas.

a) `bb := chr(i) = 32`
b) `bb := not (i-y)`
c) `bb := (sqrt(x) DIV 2) <  y`
d) `bb := ord(car) > 'A'`
e) `bb := (y <  x) or (not bb)`

Ejercicio 5 Dado el siguiente fragmento de programa:

```
VAR i : Integer; j : Char;
BEGIN
   for i:= 3 DOWNTO 0 DO
      for j:= chr(ord('A')+i) TO 'C' DO
         write(j)
END.
```

Indique cuál es su salida:

a) DCCBC
b) DCCBCABC
c) CBCABC
d) ABCBCC
e) ABCBCCDC

Ejercicio 6 Dada una función con encabezado:

```
function mm (a,b : integer) : real;
```

Indique cuál de las siguientes asignaciones es válida dentro del cuerpo de la función `mm`:

a) `mm := a + mm`
b) `mm := sqrt(a * mm)`
c) `mm := a div sqrt(b)`
d) `mm := a div b`
e) `mm := sqr(mm)`

Ejercicio 7 Dado el siguiente fragmento de programa:

```
const MAX = 8;
type arreglo : array [1..MAX] of integer;
var  z : arreglo;

procedure cualquiera(var a : arreglo);
var i : integer;
begin
   i:= 1;
   while (i <= MAX) and (a[i] = a[MAX-i+1]) do
   begin
     a[i]:= 2 * a[i];
     i:= i + 1
   end
end;
```

En el programa principal el subprograma `cualquiera` es invocado con parámetro efectivo `z`. Luego de la invocación `z` queda con los siguientes valores:

2	4	6	8	4	3	2	1

Indique cuál de los siguientes **no** puede ser el estado de `z` antes de la invocación:

a)

2	4	6	8	4	3	2	1

b)

1	4	6	8	4	3	2	1

c)

1	4	6	4	8	3	2	1

d)

1	2	3	4	4	3	2	1

e)

1	2	3	8	4	3	2	1

Ejercicio 8 Para n = 0 indique cuál de las expresiones siguientes **no** da un error en tiempo de ejecución.

a) `(n = 0) and (m div n > 0)`
b) `(n = 0) or (m div n > 0)`
c) `(m div n > 0) and (n = 0)`
d) `(m div n > 0) or (n = 0)`
e) `(m div n = m div n) and (n = 0)`

Ejercicio 9 Dado el siguiente programa:

```
program ejpuntos;
const N    = 6;
      CRUZ = 'x';
var cruces : array [1..N] of integer;
    i,j    : 1..N;
begin
  for i := N div 2  downto 1 do
    cruces[i] := 2 * i;
  for i := N div 2 downto 1 do
  begin
    for j := cruces[i] downto 1 do
      write(CRUZ);
    writeln
  end
end.
```

Indique cuál es su salida:

a)
```
xx
xxxx
xxxxxx
```

b)
```
xxxxxx
xxxx
xx
```

c)
```
xxxxxx
xxxx
xx
x
```

d)
```
xxx
xx
x
```

e) No tiene salida.

Ejercicio 10 Dado el siguiente programa:

```
program cuatro;
var w : Integer;

procedure bla (var x : Integer; y : Integer);
var z : Integer;
begin
  z := y + x;
  x := x + z;
end;

begin
  w := 3;
  bla (w, w+1);
  writeln (w);
end.
```

Indique cuál es su salida.

a) `3`
b) `4`
c) `7`
d) `10`
e) `11`

Ejercicio 11 Dado el siguiente programa.

```
program char_ordinals;
var c       : char;
    oc, bin : integer;
begin
  read(c);
  oc := ord(c);
  bin := (oc - ord('0')) mod 2;
  if (bin <> 0) and (oc div bin = oc) then
     writeln('SI')
  else
     writeln('NO')
end.
```

Indique cuál de las siguientes afirmaciones es verdadera:

a) Si de la entrada estándar se lee el caracter '9' o el caracter '7' en la salida estándar se despliega `NO`.
b) No importa cual sea el caracter leído, en la salida estándar siempre se despliega `SI`.
c) Si se usa circuito corto para evaluar las expresiones booleanas la ejecución nunca da un error. En cambio, si se usa circuito completo da error en algunos casos.
d) La ejecución nunca da un error, independientemente de si se usa circuito corto o circuito completo.
e) Si se usa circuito completo para evaluar las expresiones booleanas la ejecución nunca da un error. En cambio, si se usa circuito corto da error en algunos casos.

Ejercicio 12 Sean las variables `i1`, `i2` e `i3` de tipo `integer` y la variable `b` de tipo `boolean`.

Dado el siguiente fragmento de programa:

```
if i1 >= i2 then
  b := i2 = i3
else
  b := i1 < i2
```

Indique qué expresión equivale al valor que tiene `b` luego de ejecutar el `if`:

a) `(i1 >= i2) or (i1 < i2)`
b) `(i1 >= i2) or (i2 = i3) and (i1 < i2)`
c) `(i1 < i2) or (i2 = i3)`
d) `true`
e) `(i1 >= i2) and ((i2 = i3) or (i1 < i2))`

Ejercicio 13 Dado el siguiente programa:

```
program prodNat;
var a,b,prod : integer;
    nat      : boolean;

function multpos (a,b: integer): integer;
var i, prod: integer;
begin
  prod := 0;
  for i:= 1 to b do
    prod := prod + a;
  multpos := prod
end;

procedure multN (a,b: integer; var prod:integer;
                 var nat:boolean);
begin
  nat := true;
  if (a >= 0) and (b >= 0)
  then prod := multpos(a,b)
  else
  begin
   prod := 0;
   nat  := false
  end
end;

begin
   readln(a,b);
   multN(a,b,prod,nat);
   if nat then writeln(prod)
end.
```

Indique cuál de las siguientes afirmaciones es verdadera:

a) Si `a` y `b` son naturales, se despliega el resultado de su multiplicación
b) Si el valor de la variable `nat` es `false` como resultado de la invocación `multN(a,b,prod,nat)` entonces `a` y `b` son ambos enteros negativos
c) Si el valor de la variable `nat` es `false` como resultado de la invocación `multN(a,b,prod,nat)` entonces el programa da error en tiempo de ejecución
d) Sin importar la entrada, el programa no despliega nada
e) Si el valor de la variable `prod` es 0 como resultado de la invocación de `multN(a,b,prod,nat)` entonces el valor de `a` es 0

Ejercicio 14 Dada la siguiente definición de función:

```
function multiploK (a:integer) : boolean;
{ PRECONDICION : a >= 0 }
var n, m : integer;
begin
   m := a mod 2;
   n := a;
   while n > 0 do
     n := n - 3;
   multiploK := (m = 0) and (n = 0)
end;
```

Indique en qué caso la función devuelve `true`.

a) Nunca, porque siempre da error al ejecutar.
b) Siempre.
c) Cuando `a` es múltiplo de dos.
d) Cuando `a` es múltiplo de tres.
e) Cuando `a` es múltiplo de seis.

Ejercicio 15 Indique cuál de los siguientes fragmentos de código calcula correctamente la suma de todos los dobles de los números enteros entre dos valores `inferior` y `superior` (`inferior < superior`) incluyendo a ambos. Por ejemplo: si `inferior` es 5 y `superior` es 8, el valor de `sumadobles` es 5*2 + 6*2 + 7*2 + 8*2.

a)
```
FOR i := inferior TO superior DO
    sumadobles := sumadobles + i*2
```

b)
```
sumadobles := 0;
FOR i := superior TO inferior DO
    sumadobles := sumadobles + i*2
```

c)
```
sumadobles := inferior;
FOR i := inferior+1 TO superior DO
    sumadobles := sumadobles + i*2
```

d)
```
sumadobles := 0;
FOR i := inferior TO superior DO
    sumadobles := sumadobles + i;
sumadobles := sumadobles*2
```

e)
```
sumadobles := 0;
FOR i := inferior TO superior DO
begin
    sumadobles := sumadobles + i*2;
    i := i + 1;
end
```

Ejercicio 16 Dado el siguiente programa:

```
program bla;
var valor, dato, i : integer;
begin
  valor := 0;
  i := 1;
  read(dato);
  repeat
    i := i + 1;
    valor := valor + dato;
    read(dato);
  until (dato > 8) or (i > 3);
  writeln(valor);
end.
```

Si la entrada es:

```
3 5 6 11 2
```

Indique cuál es su salida.

a) 11
b) 18
c) 14
d) 22
e) Ninguno de los anteriores.

Ejercicio de Resolución

- El puntaje máximo por este ejercicio es **8 puntos** (no resta puntos).
- Este ejercicio debe resolverse en esta hoja.
- En este ejercicio se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera se tendrá en cuenta entre otros conceptos: indentación apropiada, correcta utilización de las estructuras de control, código elegante y legible, eficiencia de los algoritmos, etcétera.

Dado la siguiente declaración de arreglo de caracteres:

```
const N = ... ; {algun valor mayor que 1}
type arreglo = array [1..N] of char;
```

Definir una función:

```
function palindromo ( arr : arreglo ) : boolean;
```

que verifica si el contenido de un `arreglo` es un palíndromo. Un palíndromo es una palabra o frase que es igual si se lee de izquierda a derecha que de derecha a izquierda. Por ejemplo, son palíndromos:

Con N = 17:

'j'	'e'	'l'	'e'	'n'	'o'	'v'	'i'	'p'	'i'	'v'	'o'	'n'	'e'	'l'	'e'	'j'

Con N = 14:

'n'	'e'	'v'	'e'	'r'	'o'	'd'	'd'	'o'	'r'	'e'	'v'	'e'	'n'

Respuesta:

```
function palindromo ( arr : arreglo ) : boolean;
var i, medio : integer;
begin
  i := 1;
  medio := N div 2;
  while (i <= medio) and (arr[i] = arr[N-i+1]) do
    i := i + 1;
  palindromo := i > medio
end;
```

Respuestas

[1] a
[2] d
[3] c
[4] e
[5] c
[6] d
[7] c
[8] b
[9] b
[10] d
[11] c
[12] c
[13] a
[14] e
[15] d
[16] c