PRÁCTICA – EXPRESIONES Y ENUNCIADOS

1. Defina los conceptos de precedencia y asociatividad de un operador.
2. Defina efecto lateral. Dé ejemplos.
3. Defina operador sobrecargado y expresión de modo mixto.
4. Qué es coerción? Defina conversión por estrechamiento y por ensanchamiento.
5. ¿Qué es una evaluación de corto circuito? Nombre lenguajes que la usan para las expresiones booleanas, lenguajes que no la usan y algún lenguaje en el que el programador pueda elegir.
6. ¿Cómo soporta C las expresiones relacionales y booleanas?
7. ¿Cuál es el propósito de un operador de asignación? Indique una posible desventaja de tratar un operador de asignación como si fuera un operador aritmético.
8. ¿Cuál es la asociatividad de un operador unario en C?
9. Java estipula que todas las expresiones, incluyendo los argumentos de llamadas, deben ser evaluadas de izquierda a derecha. ¿Cuál es la razón de esto? ¿Por qué no sucede esto en C y en C++?
10. Asuma la siguiente regla de asociatividad y precedencia de las expresiones:

Mayor precedencia	`*,/,not`
	`+,-,&,mod`
	`-(unary)`
	`=, /=, <,<=,>,>=`
Menor precedencia	`or, xor, and`

Asociatividad: Izquierda a derecha .
Analice el orden de evaluación de las siguientes expresiones separando con paréntesis las subexpresiones y colocando un índice superior en el paréntesis derecho para indicar el orden.

Por ejemplo, en la expresión:

```
a+b*c+d
```

la evaluación del orden sería:

$((a+(b*c)^{1})^{2}+d)^{3}$

a. `a*b-1+c`
b. `a*(b-1)/c mod d`
c. `a>b xor C or d <=17`
d. `–a + b`

11. Resuelva el ejercicio anterior asumiendo que no hay reglas de precedencia y que todos los operadores se asocian de izquierda a derecha.
12. Dada la función fun definida como:

```
int fun (int *k) {
      *k +=4;
      return 3 * (*k)-1;     }
```

Suponga que fun se usa en el siguiente programa escrito en C:

```
void main(){
      int i=10, j=10, sum1, sum2;
      sum1= (i/2)+fun(&i);
      sum2= fun(&i) +(i/2);   }
```

Determine los valores de Sum1 y Sum2. Pruebe este programa en una computadora. ¿Da el valor esperado? Explique los resultados.

13. Dadas las siguientes funciones:

```
int cube (int x) {return x*x*x;}
int sum (int x, int y, int z) {return x + y +z ;}
```

Describa el proceso de evaluación de orden normal en la expresión `sum(cube(2), cube(3),cube(4))` y compárelo con la de orden aplicativo. ¿Cuántas sumas y multiplicaciones se realizan en cada caso?

14. Considere el siguiente programa C:

```
int fun (int *i) {
      *i +=5;
      return 4;
      }
void main () {
      int x=3;
      x=x+fun(&x);
      }
```

¿Cuál es el valor de x al finalizar main,
   a. si los operandos son evaluados de izquierda a derecha?
   b. si los operandos son evaluados de derecha a izquierda?

15. ¿Qué diferencia hay entre estructura de control y sentencia de control?
16. ¿Qué es un bloque?
17. ¿Cuáles son los tópicos de diseño para las estructuras de selección?
18. ¿Cuáles son las soluciones comunes para el problema de anidamiento de los selectores de dos vías?
19. Describa las sentencias de selección disponibles en Fortran I y Fortran 90.
20. ¿Cuáles son los tópicos de diseño de los enunciados de selección múltiple?
21. Compare la selección múltiple de C con la de Pascal.
22. ¿Cuáles son los tópicos de diseño de los enunciados loop controlados por contador? Compare el for de Pascal, de C y de Algol 60.
23. ¿Cuáles son los tópicos de diseño del enunciado loop controlado por condición?
24. Compare los bucles controlados por condición disponibles en C, Pascal y Ada.
25. Mencione y describa distintos mecanismos de control de bucle. ¿Cuáles están disponibles en C, Java, Ada y Pascal?
26. ¿Por qué se conserva la bifurcación incondicional en lenguajes de programación estructurados?
27. ¿Qué son los comandos protegidos?
28. ¿Qué es “pattern matching”?
29. ¿Cómo resuelve Smalltalk los enunciados condicionales?
30. ¿Qué mecanismos de iteración se pueden utilizar en Smalltalk? ¿A qué clase corresponden los mensajes que permiten la iteración controlada por contador? ¿Y la controlada por condición?
31. ¿Cómo se manejan las estructuras condicionales en Prolog? ¿Y en Haskell?
32. ¿Cómo se resuelven las estructuras repetitivas en Prolog? ¿Y en Haskell?
33. ¿Qué es un control de iteración definido por el usuario?
34. Reescriba y pruebe en Pascal, Fortran 90 (Sin Probar), Ada, C, Smalltalk y Java el siguiente segmento de código usando una estructura de repetición:

```
k:= (j+13) / 27
Loop:
   If k > 10 then goto Out
   k:= k+ 1
   i:= 3 + k-1
   Goto Loop
Out:....
```

Asuma que todas las variables son de tipo entero. Discuta qué lenguajes, para este código, tienen la mejor escritura, la mejor legibilidad, y la mejor combinación ambas.

35. Dados los siguientes enunciados, analice si son correctos en C, C++ y Java. Para aquellos que son correctos indique cuál es el valor de x:

`int x=2;`
`switch(x) x++;`
	`int x=2;`
	`switch(x) {x++;}`
		`int x=2;`
		`switch(x) {`
		`  case 1: if (x>2)`
		`  case 2: x++;`
		`  default: break;`
		`}`
			`int x=2;`
			`switch(x % 4) {`
			`  case 0: if(x >2){ x++;`
			`  case 3:  x++;`
			`  case 2:  x++;`
			`  case 1:  x++;`
			`  }`
			`}`

36. Analice si los siguientes enunciados son equivalentes:

```
int i = 0;
while (a[ i++] != 0)
   ;
```

```
for( i = 0 ; a[i] != 0 ; i++)
    ;
```

¿Con qué valor queda la variable i en cada caso?

-

37. ¿Qué es una excepción?
38. ¿Qué es un aserto? ¿Cuándo se produce una violación de un aserto?
39. ¿Qué ocurre cuando un lenguaje no provee manejo de excepciones? ¿Qué soluciones hay para el tratamiento de errores?
40. ¿Cuáles son los tópicos de diseño para manejo de excepciones?
41. ¿Qué significa que una excepción sea cargada al manejador de excepciones?
42. Compare el manejo de excepciones en Java, C++, Ada, PL/1, CLU, ML y Eiffel. Tópicos a tener en cuenta:
    - Modelo o criterio.
    - Dónde se colocan los manejadores.
    - Cómo se declaran.
    - Cómo se ligan los manejadores.
    - Alcance.
    - Cómo se levantan las excepciones.
    - Otras características propias del lenguaje.
43. Desarrolle una clase o un TAD Fecha con las siguientes características:
    a. Una fecha se crea con tres enteros día, mes y año.
    b. A una fecha se le puede consultar su día, mes o año.
    c. A una fecha se le puede modificar su día, mes o año.
    d. A una fecha se le puede consultar la cantidad de días de diferencia con otra fecha siendo el resultado positivo si es posterior a la otra y negativo en caso contrario.
    e. A una fecha se le puede solicitar la fecha que resulte al adicionarle X días, X recibido como parámetro, donde X está entre –60 y 360.

    Contemple manejo de excepciones para asegurar que:
    - Se trata de una fecha válida.
    - El día se encuentra entre 1 y el último día del mes correspondiente.
    - El mes se encuentra entre 1 y 12.
    - El año está entre 1900 y 2999.
    - X es un valor entre –60 y 360.

    a. Escriba y pruebe el código en Java.
    b. ¿Cómo manejaría las situaciones excepcionales en lenguaje C? ¿Y en Pascal?
    c. ¿Cómo lo resolvería en Smalltalk?
44. Suponga que un bloque try en Java tiene un código que quisiera ejecutar a la salida, sin importar si ha ocurrido una excepción o no. ¿Dónde tendría que definir tal código? ¿Qué pasaría si el código tuviera que ejecutarse sólo si no ocurriera la excepción?
45. Cuál es el error del siguiente código Java:

```
try{
 ...
} catch (Exception e) {...}
catch (ExceptionB e) {...}
```

46. Dado el siguiente esquema de operaciones P1, P2, P3 y main, indicar:
    a. Qué sucede si en P1 ocurre la condición A.
       Explique cómo se ubica el manejador de excepciones y en qué sentencia continúa luego de manejar la excepción, tanto para el modelo de terminación como de reasunción. Indique los nro. de las sentencias que se ejecutarían en P3.
    b. Ídem si P1 se ejecuta normalmente pero en P2 ocurre la condición B.

```
P1
If (A) throw E1
If (B) throw E2
C
```

```
P2
If (B) throw E3
D
```

```
P3
Try{
      P1
      P2
}Catch( E1){ …}
}Catch( E2){ …}
finally{ E }
F
```

```
main
Try{
      P3
}Catch( E3){ …}
finally {P2}
G
```