Programación I: Cuestiones cortas

Facultad de Estudios Estadísticos
Universidad Complutense de Madrid
Curso 2019-2020

Ejercicio 1 Determina, dados dos números naturales `n` y `p`, el valor de la expresión booleana siguiente.

```
(n%p==0)&&(n%p==n)
```

Ejercicio 2 Determina, dado un número natural `n`, la condición que se satisface cuando se ejecuta `bloque de instrucciones 2`.

```
if ((n%2!=0)||(n%3!=0)) {
  /* bloque de instrucciones 1 */ }
else {
  /* bloque de instrucciones 2 */ }
```

Ejercicio 3 Determina, dado un número natural `n`, la condición de parada del bucle siguiente.

```
while ((n>0)&&(continuar)) {
  /* bloque de instrucciones */
}
```

Ejercicio 4 Determina, dado un número natural `n`, la condición de parada del bucle siguiente.

```
while ((n>=1)&&(not parar)) {
  /* bloque de instrucciones */
}
```

Ejercicio 5 Determina, dados dos números naturales `n` y `p`, la condición de parada del bucle siguiente.

```
while ((n>0)||(p%2!=0)) {
  /* bloque de instrucciones */
}
```

Ejercicio 6 Determina, dado el prototipo de función

```
bool existeSolEcPrimerGrado(float a, float b);
```

si las siguientes invocaciones son o no correctas.

1. `existeSolEcPrimerGrado(3.0,5.0);`
2. ```
   if (existeSolEcPrimerGrado(3.0,5.0)) {
     /* bloque de instrucciones */
   }
   ```
3. `bool haySolucion=existeSolEcPrimerGrado(2*3.0,5.0);`
4. ```
   float coeficienteA; float coeficienteB;
   bool haySolucion=existeSolEcPrimerGrado(coeficienteA,coeficienteB);
   ```
5. ```
   float coeficienteA=3.0; float coeficienteB=5.0;
   bool haySolucion=existeSolEcPrimerGrado(coeficienteA,coeficienteB);
   ```

Ejercicio 7 Determina, dado el prototipo de función

```
int solEcPrimerGrado(float a, float b, float &sol);
```

si las siguientes invocaciones son o no correctas.

1. `float solucion=solEcPrimerGrado(3.0,5.0);`
2. `int cantidadSoluciones=solEcPrimerGrado(3.0,5.0,-5/3);`
3. `int cantidadSoluciones=solEcPrimerGrado(3.0,5.0,-5.0/3.0);`
4. ```
   float solucion=0;
   solEcPrimerGrado(3.0,5.0,solucion);
   ```
5. ```
   float solucion=0;
   int cantidadSoluciones=solEcPrimerGrado(3.0,5.0,solucion);
   ```
   ```

Ejercicio 8 Determina, dado el prototipo de función

```
void existeSolEcPrimerGrado(float a, float b, bool &haySol);
```

si las siguientes invocaciones son o no correctas.

1. `existeSolEcPrimerGrado(3.0,5.0,bool haySolucion);`
2.
```
float coeficienteA=3.0; float coeficienteB=5.0;
existeSolEcPrimerGrado(coeficienteA,coeficienteB,true);
```
3.
```
bool haySolucion;
haySolucion=existeSolEcPrimerGrado(3.0,5.0);
```
4.
```
bool haySolucion=false;
cout << existeSolEcPrimerGrado(3.0,5.0,haySolucion);
```
5.
```
bool haySolucion=false;
existeSolEcPrimerGrado(3.0,5.0,haySolucion);
```

Ejercicio 9 Determina, dados los prototipos de funciones

```
float suma(float a, float b);
void resta(float a, float b, float &sol);
```

si las siguientes invocaciones son o no correctas.

1. `resta(7.5,5.0,suma(3.0,-0.5));`
2.
```
float solucion=0;
resta(7.5,suma(3.0,5.0),solucion);
```
3.
```
float solucion=0;
float resultado=suma(3.0,resta(7.5,5.0,solucion));
```
4.
```
float solucion=0;
resta(7.5,5.0,solucion);
float resultado=suma(3.0,solucion);
```
5.
```
float solucion=suma(3.0,5.0);
resta(7.5,solucion,solucion);
```

Ejercicio 10 Determina, dados los prototipos de funciones

```
float suma(float a, float b);
void resta(float a, float b, float &sol);
```

si las siguientes instrucciones son o no correctas.

1. `int resto=suma(3.0,5.0)%suma(6.0,10.0);`
2. `if (suma(3.0,5.0)/suma(4.0,6.0)>0);`
3.
```
float solucion=0;
if (suma(3.0,5.0)==resta(10.0,2.0,solucion)) {
  /* bloque de instrucciones */
}
```
4.
```
float x=10.0; float y=1.0; float solucion=0;
while (resta(x,y,solucion)>0) {
   x=x-1; y=y+1;
}
```
5.
```
float x=10.0; float y=1.0;
float solucion=0; resta(x,y,solucion);
while (solucion>0) {
   x=x-1; y=y+1;
}
```