2020-2021 Evaluación. Convocatoria 27 de enero:

-Pregunta 1.

Considere un sistema de ficheros de Linux como el que se muestra a continuación, en el que el directorio proyecto-examen es un directorio “hijo” del directorio “home” del usuario.

```
proyecto-examen
├── build
│   └── CMakeFiles
│       ├── 3.13.4
│       │   └── CompilerIdCXX
│       │       └── tmp
│       ├── CheckLibraryExists
│       ├── CMakeTmp
│       ├── runTests.dir
│       │   ├── src
│       │   └── test
│       └── user_program.dir
│           └── src
├── src
└── test
```

Suponga que su directorio de trabajo (en el que se encuentra trabajando) es el directorio build.

En esta situación, y sin modificar el directorio en que se encuentra (no utilice el comando cd), escriba 10 líneas numeradas en cada una de las cuales figure el comando Linux (un único comando por línea) que podría usar para realizar las siguientes tareas:

1. Determinar cuál es el directorio en que se encuentra. **pwd**

2. Borrar todos los ficheros del directorio src cuyo nombre acaba con el carácter ~
   rm ../src/*~

3. Copiar todos los ficheros del directorio test con extensión .cc a un directorio llamado copia que “es hijo” del directorio home del usuario.
   cp ../test/*.cc /home/usuario/copia

4. Copiar el fichero CMakelists.txt que se encuentra en el directorio padre del actual al directorio /tmp del sistema de ficheros. **cp ../CMakelists.txt /tmp**

5. Determinar cuál de los ficheros con código fuente de C++ (extensión .cc) que se encuentran en el directorio src contiene la función main().
   grep -l “main()” ../src/*.cc

6. Listar (en formato largo) todos los ficheros que se encuentran en el directorio actual haciendo que se muestren al final los ficheros que han sido modificados más recientemente. **ls -ltr**

7. Determinar el tipo de fichero (texto, ejecutable, directorio, ...) de todos los ficheros y directorios existentes en el directorio actual de trabajo, que es build. **file ***

8. Borrar recursivamente todos los ficheros y directorios descendientes del directorio actual. **rm -rf ***

9. Crear en el directorio “home” del usuario un fichero comprimido (con gzip) que contenga todos los ficheros (y directorios) descendientes del directorio proyecto-examen. Tiene libertad para elegir el nombre de ese fichero, y la extensión que elija para el mismo debiera ser coherente con el contenido que va a tener.
   zip /home/usuario/comprimido.gzip ../*
   “este me lo inventé un poco xD, no se fien mucho”

10. Ejecute un comando que le permita conocer el tamaño en bytes del fichero que creó en el paso anterior. **ls -l /home/usuario/comprimido.gzip**

-Pregunta 2.

Paso de parámetros a una función en C++.

Explique (dos caras de un folio escrito a mano, a lo sumo) las formas de paso de parámetros, las implicaciones y uso de cada una de ellas.

Ponga un ejemplo de cada una.

-Pregunta 3.

El cuadrado de la suma de los 10 primeros números naturales es (1 + 2 + ... + 10)^2 = 55^2 = 3025.

La suma de los cuadrados de los primeros diez números naturales es 1^2 + 2^2 + ... + 10^2 = 385.

Por tanto la diferencia entre el cuadrado de la suma y la suma de cuadrados es 3025 - 385 = 2640.

a) Escriba todo el código que debiera figurar en un fichero diferencia_de_cuadrados.cc

```
#include<iostream>
#include<cmath>

double CuadradoSuma(int n){
  double suma = 0;
  for(int i = 1; i <= n; i++){
    suma += i;
  }
  suma = pow(suma, 2);
  return suma;
}

int SumaCuadrados(int n){
  int cuadrado = 0;
  for(int i = 1; i <= n; i++){
    cuadrado += pow(i, 2);
  }
  return cuadrado;
}

int Diferencia(int n){
  return CuadradoSuma(n) - SumaCuadrados(n);
}
```

El código que escriba debe incluir el de tres funciones que calculen el cuadrado de la suma, la suma de cuadrados y la diferencia entre los valores anteriores para los números naturales en el rango [1, N] siendo N un valor que se pasa como parámetro a dichas funciones.

b) Escriba también el código que debiera figurar en un fichero diferencia_de_cuadrados_test.cc que contenga al menos un par de (Google) tests unitarios que evalúen el correcto funcionamiento de las funciones del apartado anterior.

-Pregunta 4.

A continuación puede encontrar la definición parcial de la clase FootballTeam en C++, representa un equipo de fútbol y que figura en un fichero football_team.h:

```
class FootballTeam {
 public:
  FootballTeam(std::string nombre, int puntos, int goles) : name_(nombre),
points_(puntos), goals_(goles) {}
  friend std::istream& operator>>( std::istream& in, const FootballTeam&);
  std::string name() { return name_; }
  int points() { return points_; }
  int goals() { return goals_; }
 private:
  std::string name_;
  int points_;
  int goals_;
};
```

Complete la declaración de la clase FootballTeam añadiendo:

-Los constructores que considere oportunos.
-Implemente la sobrecarga del operador de inserción (<<)
-Implemente la sobrecarga de operadores requerida para comprobar que un equipo se encuentra por delante de otro en la tabla clasificatoria. Un equipo se encuentra por delante de otro en la tabla clasificatoria si (i) tiene mayor número de puntos o, (ii) teniendo el mismo número de puntos, ha marcado mayor cantidad de goles.

Una vez realizada la implementación de las funcionalidades expuestas, implemente una función main donde se instancien 2 objetos de tipo FootballTeam con un número de puntos y un número de goles diferente y muestre por pantalla cuál de los dos se encuentra mejor clasificado.

-Pregunta 5.

Es un cuestionario, no lo tengo :(

2020-2021 Evaluación. Convocatoria 4 de febrero

-Pregunta 1.

Considere un sistema de ficheros de Linux como el que se muestra a continuación, en el que el directorio proyecto-examen es un directorio “hijo” del directorio “home” del usuario.

proyecto-examen

```
├── build
│   └── CMakeFiles
│       ├── 3.13.4
│       │   └── CompilerIdCXX
│       │       └── tmp
│       ├── CheckLibraryExists
│       ├── CMakeTmp
│       ├── runTests.dir
│       │   ├── src
│       │   └── test
│       └── user_program.dir
│           └── src
├── src
└── test
```

Suponga que su directorio de trabajo (en el que se encuentra trabajando) es el directorio build.

En esta situación, y sin modificar el directorio en que se encuentra (no utilice el comando cd), escriba 10 líneas numeradas en cada una de las cuales figure el comando Linux (un único comando por línea) que podría usar para realizar las siguientes tareas:

1. Determinar cuál es el directorio en que se encuentra. **pwd**
2. Borrar todos los ficheros del directorio build/src cuyo nombre acaba con el carácter ~ **rm ../src/*~**

3. Copiar todos los ficheros del directorio test con extensión .cc a un directorio llamado copia que “es hijo” del directorio home del usuario.
   cp ../test/*.cc /home/usuario/copia

4. Copiar el fichero CMakelists.txt que se encuentra en el directorio padre del actual al directorio /tmp del sistema de ficheros. **cp ../CMakelists.txt /tmp**

5. Comprobar que la dirección 193.71.201.124 está disponible. **ping 193.71.201.124**

6. Listar (en formato largo) todos los ficheros que se encuentran en el directorio actual haciendo que se muestren al final los ficheros que han sido modificados más recientemente. **ls -ltr**

7. Determinar el tipo de fichero (texto, ejecutable, directorio, ...) de todos los ficheros y directorios existentes en el directorio actual de trabajo, que es build. **file ***

8. Borrar recursivamente todos los ficheros y directorios descendientes del directorio actual. **rm -rf ***

9. Muestre el contenido del fichero CMakelists.txt. **cat ../CMakelists.txt**

10. Ejecute un comando que le permita conocer el tamaño en bytes del fichero CMakelists.txt. **ls -l ../CMakelists.txt**

-Pregunta 2.

Cuando se pasa un objeto como parámetro a una función, una buena práctica es hacerlo mediante un parámetro que sea una referencia constante.

Justifique (no más de medio folio de texto escrito) la razón de esta recomendación.

Escriba el prototipo de una función que contemple un único parámetro pasado como referencia constante.

-Pregunta 3.

La función exponencial, f(x)=e^x, se puede calcular mediante su desarrollo en serie de Taylor:

desarrollo en serie de Taylor

Cuantos más términos tome de la serie, mayor será la precisión de la aproximación a la función.

Diseñe una función

double MyExp(double exponent, unsigned num_terms);

que calcule el valor de e^x para un valor de x que se le pase como parámetro (exponent). El segundo parámetro de la función indica el número de términos del desarrollo en serie de Taylor a utilizar para el cálculo.

Su función debería utilizar otra función que calcule el factorial de un número que se le pase como parámetro.

Compare los valores que obtiene en su función para diferentes valores del parámetro y compárelos con los que obtiene al usar la función exp definida en el fichero de cabecera cmath.

```
#include<iostream>
#include<cmath>

double Factorial(int number){
   switch (number){
   case 0:
   case 1:
      return 1;
   default:
      double factorial = 1.0;
      for(int i = 1; i <= number; i++){
         factorial *= i;
      }
      return factorial;
   }
}


double Power(double base, int exponente){
   double resultado = 1.0;
   if(exponente == 0){
      return 1.0;
   }
   for(int i = 0; i < exponente; i++){
      resultado *= base;
   }
   return resultado;
}
```

```
double MyExponencial(const double exponente, const int num_terms){
    double sum = 1.0;
    for(int i = 1; i< num_terms; i++){
        sum += Power(exponente, i) / Factorial(i);
    }
    return sum;
}
```

-Pregunta 4.

A continuación puede encontrar la definición parcial de la clase Student en C++, la cual representa un estudiante:

```
class Student {
 public:
  std::string name() { return name_; }
  std::string surname() { return surname_; }
  int grade() { return grade_; }
  friend std::ostream& operator<<(std::ostream& os, const Student& pepe);
  Student Operator=(const Student pepe);
 private:
  std::string name_;
  std::string surname_;
  int grade_;
};
std::ostream& operator<<(std::ostream& os, const Student& pepe){
  os << “Nombre: “ << pepe.name() << “\nApellido: “ << pepe.surname() << “\nNota: “
   < < pepe.grade() << std::endl;

   return os;
 }
```

Complete la declaración de la clase Student añadiendo:

-Los constructores que considere oportunos.

-Implemente la sobrecarga del operador de inserción (<<)
-Implemente la sobrecarga de operadores requerida para comprobar si un estudiante está antes que otro en el listado ordenado de calificaciones de mayor a menor.

Una vez realizada la implementación de las funcionalidades expuestas, implemente una función main donde se instancien 2 objetos de tipo Student con calificaciones diferentes y muestre por pantalla cuál de los dos se encuentra antes en el listado de calificaciones.

-Pregunta 5.

Empareje cada término con su definición

Una fila de una relación	Atributo ✖
Una columna de una relación	Relación ✖
Una tabla con filas y columnas	Tupla ✖
Conjunto de valores que puede tomar un atributo	Dominio ✔

Indique cuál es el resultado que imprime en pantalla este programa:

```
#include <iostream>

int Compute(int x, int y) {
  int result{0};
  while (y != 0) {
    result = result + x;
    y = y - 1;
  }
  return(result);
}

int main() {
  int x{5}, y{5};
  std::cout << Compute(x, y) ;
  return(0);
}
```

- (●) El programa produce un error en tiempo de ejecución ✖
- ( ) 25
- ( ) 10
- ( ) 35
- ( ) 30
- ( ) 20

Pregunta 3

Parcialmente correcta

Puntúa 0,50 sobre 1,00

Indique los que son ejemplos de sistemas operativos monolíticos

- [ ] OS/2
- [ ] Fortran
- [ ] LibreOffice
- [ ] Cobol
- [x] GNU/Linux ✔
- [ ] MS-DOS

¿Cuál es la salida del siguiente programa?:

```
include <iostream>

int Func (int x = 0, int y = 5, int z) {
  return (x + y + z);
}

int main () {
  std::cout << Func(10);
  return 0;
}
```

- [ ] 0
- [ ] 5
- [ ] Da un error semántico
- [ ] 10
- [x] No compila porque da un error sintáctico ✔
- [ ] 15

Indique la función que realiza cada uno de los siguientes comandos Linux:

`ping`	Determinar si otro ordenador es accesible desde la red o no	✔
`sftp`	Transferir ficheros de forma segura de un ordenador origen a otro destino	✔
`file`	Determinar el tipo de un fichero	✔
`df`	Mostrar la cantidad de disco libre en un sistema de ficheros	✔
`tar`	Crear un único fichero que almacene todo el contenido de un directorio	✔
`grep`	Buscar una cadena de texto en un fichero	✔

Indique la relación existente entre una clase que use el método std::cout y la clase std::ostream

- ○ Composición
- ○ Asociación
- ○ Herencia
- ○ Dependencia
- ◉ No tiene porqué haber relación alguna entre esas clases ✖
- ○ Agregación

Pregunta 7

Parcialmente correcta

Puntúa 0,50 sobre 1,00

Supongamos que tenemos una declaración como esta

```
vector<string> capital_country = {"Tokio", "Japon"};
```

quisiera añadir una cadena más al vector, ¿es posible?

- ☐ Si, utilizando el método front()
- ☐ No es posible: a un std::vector no se le puede cambiar el número de elementos que almacena
- ☐ Si, utilizando el método back()
- ☑ Si, utilizando el método push_back() ✔
- ☐ Si, utilizando el método emplace_back()
- ☐ No es posible

Indique qué fragmentos de código están correctamente escritos de acuerdo a la Guía de Estilo de código que se utiliza en la asignatura

- [ ]
```
mi_presupuesto = LlamadaFuncion(0, valor_presupuestado);
```

- [ ]
```
std::string MiClase::MiFuncion (int mes,double valor_final) {
    DoSomething();
    ...
}
```

- [ ] ++ mi_variable;

- [x]
```
a=a+1;
```
✖

- [x]
```
for (int i = 0; i < kMaxIteraciones; ++i)
{
  // Cuerpo del bucle
}
```
✖

- [x]
```
while (true) {
  // cuerpo del bucle
}
```
✔

El siguiente programa implementa la sobrecarga del operador de extracción para la clase Point.

En las líneas 9 y 21 el segundo parámetro de la función operator>> no es constante.

Indique la razón por la que ese segundo parámetro no es constante.

```
1 #include <iostream>
2
3 class Point {
4  public:
5   Point(double x = 0.0, double y = 0.0, double z = 0.0) : x_{x}, y_{y}, z_{z} {
6   }
7
8   friend std::ostream& operator<< (std::ostream& out, const Point& point);
9   friend std::istream& operator>> (std::istream& in, Point& point);
10  private:
11   double x_{};
12   double y_{};
13   double z_{};
14 };
15
16 std::ostream& operator<<(std::ostream& out, const Point& point) {
17   out << "Point(" << point.x_ << ", " << point.y_ << ", " << point.z_ << ')';
18   return out;
19 }
20
21 std::istream& operator>>(std::istream& in, Point& point) {
22   in >> point.x_;
23   in >> point.y_;
24   in >> point.z_;
25   return in;
26 }
27
28 int main() {
29   std::cout << "Enter a point: \n";
30   Point point{};
31   std::cin >> point;
32   std::cout << "You entered: " << point << '\n';
33   return 0;
34 }
```

- ○ Si se declarara el segundo parámetro como constante no se podrían "encadenar" sucesivas operaciones de extracción (>>)
- ◉ Porque el objeto que se pasa como parámetro será modificado como consecuencia de la ejecución de la función ✔
- ○ Si se declarara constante, el compilador generaría un error de compilación

Pregunta 10

Incorrecta

Puntúa 0,00 sobre 1,00

Indique las afirmaciones que son correctas

- ☑ La mayoría de las redes siguen un modelo híbrido entre los extremos cliente/servidor vs. P2P ✔
- ☐ Las redes P2P aparecieron cronológicamente antes que las redes de modelo cliente/servidor
- ☐ En una red P2P todos los ordenadores actúan simultáneamente como clientes y como servidores
- ☑ Internet es un ejemplo de una red P2P ✖
- ☑ En una red de modelo cliente/servidor el cliente es el sistema que almacena la información ✖
- ☐ Una red P2P es una red descentralizada