3. Para comenzar a usar autoconf es:

a) Aconsejable usar autoscan

b) Imprescindible usar autoscan

c) Innecesario usar autoscan

d) Obligatorio usar autoreconf

4. En la programación por contrato las precondiciones:

a) se aplican solo a métodos de una clase

b) se aplican solo a funciones independientes de cualquier clase

c) Se aplican tanto a métodos de una clase como a funciones independientes de cualquier clase.

d)se aplican en el constructor de una clase

5. En un lenguaje con soporte para la programación por contrato, las precondiciones y postcondiciones:

a) se invocan cuando el programador lo indica en métodos de una clase

b) se invocan primero las postcondiciones, luego las precondiciones y finalmente el código del método.

c) se aplican en el constructor de una clase

d) se invocan primero las precondiciones, luego el código del método y al final las postcondiciones.

6. Git es un SCVs:

a) Evolucion de BitKeeper

b) Evolucion de Mercurial

c) Evolucion de Monotone

d) Influenciado por BitKeeper

7.

a) trabajar con un Makefile por cada directorio de nuestro proyecto

b) Trabajar con un único Makefile en la raíz de nuestro proyecto

c) a que los archivos pasados a la orden make llamen a Makefile

d) a crear como minimo un fichero Makefile.am si usamos Automake

8. Las autotools se componen de:

a) Solo Autoconf

b) Solo Automake

c) Solo Autoconf y Automake

d) Ninguna de las anteriores

9. La herramienta cmake:

a) Permite tener solo un fichero CMakeList.txt en el directorio raíz del proyecto y ninguno en subdirectorios del mismo.

b) No permite tener un fichero CMakeList.txt en cada subdirectorio del proyecto.

c) Nos permite tener un fichero CMakeList.txt en cada subdirectorio del proyecto.

d) Nos obliga a tener un fichero CMakeList.txt en cada subdirectorio del proyecto.

10. Gettext

a) Traduce automáticamente todas las cadenas de nuestra aplicación.

b) Necesita que las cadenas estén marcadas de forma distinta según el lenguaje al que se traducirán.

c) Lleva a cabo la traducción palabra a palabra de una frase

d) Solo traduce las cadenas marcadas de algún modo en nuestra aplicación

11. La configuración de un proyecto software se hace:

a) En función de los resultados tras una primera compilación.

b) De forma distinta para cada máquina donde vayamos a compilarlo.

c) Realizando unos test y chequeos antes de proceder a la compilación

d) De forma manual para cada máquina donde lo vayamos a compilar

12. Un proyecto configurado con Autotools

a) Solo se puede compilar en una máquina que tenga instalada el compilador con el que se configuró

b) Necesita que esté instalado autoconf en la maquina donde se va compilar

c) No se necesita que esté instalado autoconf en la máquina donde se va a compilar.

d) Debe compilarse en la misma máquina donde se ha configurado.

13. Unas herramientas como Autotools

a) Solo se pueden emplear en proyectos en lenguaje C

b) Solo se pueden emplear en proyectos en lenguaje C++

c) Solo se pueden emplear en proyectos en lenguaje ObjectiveC

d) Ninguna de las anteriores

14. Un proyecto configurado con autotools puede constar de:

a) Un único archivo llamado Makefile.am y varios config.h.in.

b) Un archivo llamado autoconf.am

c) Un archivo llamado configure.ac y varios Makefile.am

d) Un archivo llamado Makefile.am y otro configure.ac

15. Lo que conocemos como Cmake:

a) Se compone de varias herramientas

b) Está formado solo por la aplicación cmake

c) Está formado por las aplicaciones cmake y ctest

d) Está formado por las aplicaciones cdash y cmake

16. En el paso de test, el concepto Branch Coverage

a) Se refiere a la cantidad de ramas empleadas en el controlador de versiones usado.

b) Se refiere si estamos testeando la rama master

c) Se refiere la cantidad de ramas del código empleadas

d) Se refiere si estamos testeando la rama trunk

La respuesta "a" es correcta, pero se refiere a la técnica que se utiliza en los test de datos, en la que se combinan datos reales y sintéticos para comprobar que la aplicación funciona correctamente. Sin embargo, "d" se refiere a la finalidad de los test de datos, que es comprobar que la aplicación está trabajando correctamente con los datos que se le proporcionan, ya sean reales o sintéticos. Por eso esa es la opción más precisa para describir la finalidad de los test de datos.

17. En los Test de datos:

a) Usamos datos reales y sintéticos

b) Usamos solo datos sintéticos que hacen falta

c) Usamos solo datos reales que son los que verdaderamente empleará la aplicación

d) Nos dedicamos a comprobar los datos con los que trabaja la aplicación.

18. El paso de tests…

a) sirve para mostrar los fallos de nuestro proyecto

b) sirve para mostrar la ausencia de fallos de nuestro proyecto

c) Sirve para mostrarnos las partes de nuestro poryecto que deben ser mejoradas

d) no sirven para nada

19. En el GlassBox Testing:

a) Tenemos acceso a parte del código a testear

b) Tenemos acceso sólo al código del programa principal a testear

c) Tenemos acceso a todo el código a testear

d) Tenemos acceso a un módulo del código a testear

1.- Una etiqueta de versión:

- ○ Siempre ha de seguir el formato X.Y.Z.
- ○ Siempre ha de seguir el formato W.X.Y.Z.
- ○ Siempre ha de seguir el formato YYYY.MM.DD.
- ◉ Ninguna de las anteriores.

2.- Mercurial es un SCV:

- ○ Centralizado pero con capacidades de uno distribuido.
- ○ Descentralizado pero solo en sus versiones más recientes.
- ◉ Descentralizado.
- ○ Ninguna de las anteriores.

3.- Las mejoras en tiempo de recompilación con ccache:

- ○ Solo se observan si usamos un Makefile.
- ○ Solo se observan si usamos un Makefile recursivo.
- ○ Solo se observan si usamos un Makefile creado por cmake.
- ○ Ninguna de las anteriores.

4.- En un sistema de seguimiento de fallos:

- ○ Sólo podemos dar de alta informes de fallos.
- ○ Sólo podemos dar de alta peticiones de nuevas características.
- ○ Podemos dar de alta peticiones de nuevas características además de informes de fallos.
- ○ Todas las anteriores son válidas.

5.- En un sistema de seguimiento de fallos respecto a los tipos de error:

- ○ No es importante acotarlos.
- ○ Es bueno acotarlos para facilitar el trabajo del reportador de un fallo.
- ○ Es bueno permitir al usuario que invente sus propias categorías de error.
- ○ Todas las anteriores son válidas.

6.- Autoconf + Automake:

- ○ Es portable directamente a Windows.
- ○ Es portable a Windows bajo Cygwin.
- ○ No funciona en Windows.
- ○ Ninguna de las anteriores.

7.- La cantidad de líneas de código a escribir;

- ○ Puede ser un factor a tener en cuenta al estructurar un equipo de desarrollo.
- ○ Nunca puede ser un factor a tener en cuenta al estructurar un equipo de desarrollo.
- ○ Siempre será un factor a tener en cuenta al estructurar un equipo de desarrollo.
- ○ Ninguna de las anteriores.

8.- La herramienta distcc:

- ○ Tiene en cuenta de forma automática diferencias de arquitectura al compilar.
- ○ Sólo puede usarse con compiladores GCC.
- ○ Está pensada para ser usada con compiladores GCC
- ○ Todas las anteriores son válidas.

Cuál?

9.- I18N y L10N:

- ○ Son lo mismo si el lenguaje original empleado en el código es igual al lenguaje destino.
- ○ L10N es una versión simlpificada de I18N.
- ○ I18N es una versión simplificada L10N.
- ○ Ninguna de las anteriores.

10.- El empaquetado DEB:

- ○ Permite ejecutar acciones antes de instalar pero no después.
- ○ Permite ejecutar acciones después de instalar pero no antes.
- ○ No permite ejecutar acciones ni antes ni después de instalar.
- ○ Ninguna de las anteriores.

11.- Git puede:

- ○ Trabajar con sus propios repositorios y los de cualquier otro SCV.
- ○ Trabajar solo con sus propios repositorios.
- ○ Trabajar con sus propios repositorios y los de algunos otros SCV.
- ○ Todas las anteriores son ciertas.

12.- En el paso de tests un fixture:

- ○ No es necesario en ninguna circunstancia.
- ○ Siempre es necesario.
- ○ Sólo es necesario si reservamos memoria dinámica.
- ○ Ninguna de las anteriores.

13.- En el diseño por contrato en lenguaje D los invariantes:

- ○ Se deben cumplir en métodos con todo tipo de visibilidad.
- ○ Se deben cumplir en métodos con visibilidad public.
- ○ Se deben cumplir en métodos con visibilidad private.
- ○ Ninguna de las anteriores.

14.- En el diseño por contrato para poder usar post-condiciones:

- ○ Es obligatorio usar pre-condiciones.
- ○ Es obligatorio usar invariantes.
- ○ Es opcional usar pre-condiciones.
- ○ Todas las anteriores son validas.

15.- Dado un cambio a realizar en el código, la ortogonalidad del sistema:

- ○ Es inversamente proporcional al número de líneas de código implicadas.
- ○ Es inversamente proporcional al número de programadores implicados para hacerlas.
- ○ Es directamente proporcional al número de programadores implicados para hacerlas.
- ○ Ninguna de las anteriores.

16.- Un sistema de empaquetado basado en TAR+GZ:

- ○ Es igual de eficaz que uno basado en RPM.
- ○ Es igual de eficaz que uno basado en DEB.
- ○ Es menos eficaz que DEB o RPM por determinados motivos.
- ○ Ninguna de las anteriores.

17.- Los tests de caja negra:

- ○ Miden solo el porcentaje de instrucciones testeadas.
- ○ Miden solo el porcentaje de ramas del código usadas.
- ○ Miden ambos porcentajes anteriores.
- ○ Ninguna de las anteriores.

18.- La biblioteca Gettext:

- ○ Solo permite disponer de un idioma destino.
- ○ Permite tener tantos idiomas destino como sea necesario.
- ○ Requiere escribir los mensajes originales en inglés.
- ○ Requiere escribir los mensajes originales codificados en UTF-8.

19.- Autoconf necesita a Automake:

- ○ Obligatoriamente siempre.
- ○ Obligatoriamente si desarrollamos en lenguaje C.
- ○ Opcionalmente.
- ○ Ninguna de las anteriores.

20.- Las etiquetas de versión:

- ○ Solo aportan información al usuario del software.
- ○ Solo aportan información al creador del software.
- ○ No aportan información importante ni al usuario ni al creador del software.
- ○ Ninguna de las anteriores.

La cantidad de líneas de código a escribir;
- * **Puede ser un factor a tener en cuenta al estructurar un equipo de desarrollo.**
- * Nunca puede ser un factor a tener en cuenta al estructurar un equipo de desarrollo
- * Siempre será un factor a tener en cuenta al estructurar un equipo de desarrollo.
- * Ninguna de las anteriores.

Un sistema de empaquetado basado en TAR+GZ:
- * Es igual de eficaz que uno basado en RPM.
- * Es igual de eficaz que uno basado en DEB.
- * **Es menos eficaz que DEB o RPM por determinados motivos.**
- * Ninguna de las anteriores.

Las mejoras en tiempo de recompilación con ccache:
- * Solo se observan si usamos un Makefile.
- * Solo se observan si usamos un Makefile recursivo.
- * Solo se observan si usamos un Makefile creado por cmake.
- * **Ninguna de las anteriores.**

El empaquetado DEB:
- * Permite ejecutar acciones antes de instalar pero no después.
- * Permite ejecutar acciones después de instalar pero no antes.
- * No permite ejecutar acciones ni antes ni después de instalar.
- * **Ninguna de las anteriores.**

Las etiquetas de versión:
- * Solo aportan información al usuario del software.
- * Solo aportan información al creador del software.
- * No aportan información importante ni al usuario ni al creador del software.
- * **Ninguna de las anteriores.**

La biblioteca Gettext:
- * Solo permite disponer de un idioma destino.
- * **Permite tener tantos idiomas destino como sea necesario.**
- * Requiere escribir los mensajes originales en inglés.
- * Requiere escribir los mensajes originales codificados en UTF-8.

Los tests de caja negra:
- * Miden solo el porcentaje de instrucciones testeadas.
- * Miden solo el porcentaje de ramas del código usadas.
- * **Miden ambos porcentajes anteriores.**
- * Ninguna de las anteriores.

Una etiqueta de versión:
- * Siempre ha de seguir el formato X.Y.Z.
- * Siempre ha de seguir el formato W.X.Y.Z.
- * Siempre ha de seguir el formato YYYY.MM.DD.
- * **Ninguna de las anteriores.**

Autoconf + Automake:
- * Es portable directamente a Windows.
- *** Es portable a Windows bajo Cygwin.**
- * No funciona en Windows.
- * Ninguna de las anteriores.

Git puede:
- * Trabajar con sus propios repositorios y los de cualquier otro SCV.
- * Trabajar solo con sus propios repositorios.
- *** Trabajar con sus propios repositorios y los de algunos otros SCV.**
- * Todas las anteriores son ciertas.

Dado un cambio a realizar en el código, la ortogonalidad del sistema:
- * Es inversamente proporcional al número de líneas de código implicadas.
- *** Es inversamente proporcional al número de programadores implicados para hacerlas.**
- * Es directamente proporcional al número de programadores implicados para hacerlas.
- * Ninguna de las anteriores.

En un sistema de seguimiento de fallos respecto a los tipos de error:
- * No es importante acotarlos.
- *** Es bueno acotarlos para facilitar el trabajo del reportador de un fallo.**
- * Es bueno permitir al usuario que invente sus propias categorías de error.
- * Todas las anteriores son válidas.

Autoconf necesita a Automake:
- * Obligatoriamente siempre.
- * Obligatoriamente si desarrollamos en lenguaje C.
- *** Opcionalmente.**
- * Ninguna de las anteriores.

En el diseño por contrato para poder usar post-condiciones:
- * Es obligatorio usar pre-condiciones.
- * Es obligatorio usar invariantes.
- *** Es opcional usar pre-condiciones.**
- * Todas las anteriores son validas.

En el diseño por contrato en lenguaje D los invariantes:
- * Se deben cumplir en métodos con todo tipo de visibilidad.
- * Se deben cumplir en métodos con visibilidad public.
- *** Se deben cumplir en métodos con visibilidad private.**
- * Ninguna de las anteriores.

La herramienta distcc:
- * Tiene en cuenta de forma automática diferencias de arquitectura al compilar.
- * Sólo puede usarse con compiladores GCC.
- *** Está pensada para ser usada con compiladores GCC**
- * Todas las anteriores son válidas.

En un sistema de seguimiento de fallos:
- * Sólo podemos dar de alta informes de fallos.
- * Sólo podemos dar de alta peticiones de nuevas características.
- *** Podemos dar de alta peticiones de nuevas características además de informes de fallos.**
- * Todas las anteriores son válidas.

En el diseño por contrato para poder usar post-condiciones:
- * Es obligatorio usar pre-condiciones.
- * Es obligatorio usar invariantes.
- *** Es opcional usar pre-condiciones.**
- * Todas las anteriores son validas.

I18N y L10N:
- * Son lo mismo si el lenguaje original empleado en el código es igual al lenguaje destino.
- * L10N es una versión simlpificada de I18N.
- * I18N es una versión simplificada L10N.
- *** Ninguna de las anteriores.**

En un sistema de seguimiento de fallos:
- * Sólo podemos dar de alta informes de fallos.
- * Sólo podemos dar de alta peticiones de nuevas características.
- *** Podemos dar de alta peticiones de nuevas características además de informes de fallos.**
- * Todas las anteriores son válidas.

Git puede:
- * Trabajar con sus propios repositorios y los de cualquier otro SCV.
- *** Trabajar solo con sus propios repositorios.**
- * Trabajar con sus propios repositorios y los de algunos otros SCV.
- * Todas las anteriores son ciertas.

La herramienta distcc:
- * Tiene en cuenta de forma automática diferencias de arquitectura al compilar.
- * Sólo puede usarse con compiladores GCC.
- *** Está pensada para ser usada con compiladores GCC**
- * Todas las anteriores son válidas.

En el paso de tests un fixture:
- * No es necesario en ninguna circunstancia.
- * Siempre es necesario.
- * Sólo es necesario si reservamos memoria dinámica.
- *** Ninguna de las anteriores.**

Un sistema de empaquetado basado en TAR+GZ:
- * Es igual de eficaz que uno basado en RPM.
- * Es igual de eficaz que uno basado en DEB.
- *** Es menos eficaz que DEB o RPM por determinados motivos.**
- * Ninguna de las anteriores.

El empaquetado DEB:
- * Permite ejecutar acciones antes de instalar pero no después.
- * Permite ejecutar acciones después de instalar pero no antes.
- * No permite ejecutar acciones ni antes ni después de instalar.
- *** Ninguna de las anteriores.**

Los tests de caja negra:
- * Miden solo el porcentaje de instrucciones testeadas.
- * Miden solo el porcentaje de ramas del código usadas.
- * Miden ambos porcentajes anteriores.
- *** Ninguna de las anteriores.**

Mercurial es un SCV:
- * Centralizado pero con capacidades de uno distribuido.
- * Descentralizado pero solo en sus versiones más recientes.
- *** Descentralizado.**
- * Ninguna de las anteriores.

Preguntas de DCA Test 1

by Joaquín y Pavel con la colaboración de Iván

La herramienta `distcc` :

a) Usa un solo ejecutable `distcc` .

b) Emplea dos ejecutables:: `distcc` y `distccd`. Lesson 5. Punto 15 (Distcc I)

c) En realidad,, `distcc` y `distccd` son el mismo programa lanzado con opciones distintas..

d) Necesita de make para funcionar..

Los archivos `DEB`

a) Sólo pueden ser manipulados por dpkg..

b) Se pueden manipular por `TAR`. Lo hemos probado y se extraen los 3 ficheros. Lesson 4 Puntos 8 y 9

c) Se pueden manipular por `RAR`.

d) Ninguna de las anteriores.

La herramienta `make`:

a) Sólo puede lanzar trabajos de forma secuencial..

b) Puede lanzar más trabajos en paralelo que procesadores o núcleos tiene la máquina en la que se ejecuta. Lesson 5. Punto 7 y Lesson 5. Punto 17 (ultimo punto)

c) Puede lanzar tantos trabajos en paralelo como procesadores o núcleos tenga la máquina en la que se ejecuta..

d) No puede ejecutarse de manera recursiva.

La herramienta `dh_make` :

a) Crea un paquete debian a partir de un directorio.

b) Crea un paquete debian a partir de un `TGZ`.

c) *Debianiza* un código fuente a partir de un `TGZ`. Lesson 4. Punto 11 (También punto 18 y el deb se crea con dpkg-buildpackage -rfakeroot -us -uc)

d) Instala un paquete `DEB`.

La herramienta `distcc` :

a) Funciona entre máquinas conectadas en red. No aparece en los apuntes pero lo dijo en clase

b) Sólo funciona entre máquinas de la misma red..

c) No funciona si no hay conexión a la red para la compilación..

d) Dispone de un tope de máquinas a usar para compilar.

La herramienta `distcc` en modo *bombeo* :

a) Hace lo mismo que `ccache`.

b) Hace lo mismo que en modo *sencillo*.

c) Consigue tiempos de compilación mejores. Lesson 5. Punto 16

d) Necesita de `ccache` para funcionar completamente. No. Lesson 5. Punto 20

Los sistemas de empaquetado como `DEB` o `RPM` :

a) Sólo permiten instalar software fácilmente.

b) Sólo permiten desinstalar software fácilmente..

c) Tienen en cuenta dependencias entre paquetes a la hora de instalar uno. Lesson 5. Punto 7

d) Ninguna de las anteriores..

En el desarrollo de un proyecto software, la etiqueta de versión::

a) Sólo se emplea en el código entregado al usuario final del mismo..

b) Siempre tiene el formato `X.Y.Z`

c) Sólo tiene sentido para los desarrolladores del proyecto..

d) Es útil tanto a desarrolladores como a usuarios del proyecto. Lesson 2. Punto 11

La herramienta `ccache`:

a) No hace uso de varios núcleos del procesador.

b) Siempre hace uso de varios núcleos del procesador.

c) Opcionalmente hace uso de varios núcleos del procesador. La cantidad de núcleos está marcada por make, ccache solo utiliza los resultados guardados, no procesa.

d) Puede hacer uso de hasta 2 núcleos de proceso, pero no más.

En el desarrollo de un proyecto software, la rama `trunk` o `master` :

a) Siempre contendrá una versión estable del proyecto.

b) Puede contener una versión estable del proyecto. Lesson 2. Punto 5 (Sobretodo el último punto)

c) Nunca contendrá una versión estable del proyecto.

d) No se emplea nada más que en ocasiones.

La herramienta `ccache` :

a) No sobrecarga nada en el proceso de compilación

b) Añade una sobrecarga mínima al proceso de compilación. Lesson 5. Punto 10

c) Almacena sus datos en la *cache* del disco duro.

d)) Almacena sus datos en memoria virtual..

La herramienta `distcc` :

a) No hace uso de varios núcleos del procesador.

b) Siempre hace uso de varios núcleos del procesador.

c) Opcionalmente hace uso de varios núcleos del procesador. Aunque distcc se utiliza para compilación distribuida, se puede utilizar en local y con un solo job

d) Puede hacer uso de hasta 3 núcleos de proceso,, pero no más.

En el desarrollo de un proyecto software, las ramas `trunk` y `master`:

a) Son ramas especiales del SCV empleado en el proyecto.

b) Son la misma rama del proyecto, pero con diferente nombre. No es seguro. Lesson 2. Punto 4

c) Nunca podrán existir simultáneamente.

d) Se emplean indistintamente en un mismo proyecto.

En un sistema de `bugtracking` :

a) Es útil acotar los tipos de errores a reportar. Lesson 3. Punto 4

b) No debemos acotar los tipos de errores, pues se pueden quedar sin reportar fallos.

c) No debemos dejar que el usuario reporte un " *whishlist*" como un error.

d) Un desarrollador no puede cerrar un fallo si el reportador del mismo no lo permite.

La herramienta `tar` :

a) Comprime automáticamente con gzip.

b) Comprime automáticamente con bzip2.

c) Comprime automáticamente con xz.

d) Ninguna de las anteriores. Lesson 4. Punto 5

En el desarrollo de un proyecto software, las ramas de desarrollo... :

a) Sólo contienen código de la siguiente versión estable.

b) Pueden contener código de corrección de fallos de una versión estable anterior. Lesson 2. Punto 10

c) Nunca introducen fallos en el código nuevo.

d) Ninguna de las anteriores.

Para medir la ortogonalidad de un sistema…:

a) Vemos cuanta gente hay asignada a cada parte del proyecto.

b) Vemos cuanta gente permanece en su puesto cuando no hay cambios.

c) Vemos a cuanta gente afecta un cambio hecho al proyecto. En duda con la a). Lesson 1. Punto 1

d) Ninguna de las anteriores.

El formato de un paquete `DEB` :

a) Es desconocido

b) Es el de un archivo `TAR`.

c) Es el de un archivo `AR`. Lesson 4. Punto 8

d) Es el de un archivo `TAR.GZ`.

Un sistema ortogonal se consigue…:

a) Siempre,, sin hacer nada especial.

b) Separando las distintas capas que lo componen.

c) Eliminando las capas necesarias que lo componen. Lesson 1. Punto 1

d) Ninguna de las anteriores.

La herramienta `make` cuando se ejecuta sin parámetros…:

a) Trata de obtener el primer objetivo del fichero `Makefile.`
https://stackoverflow.com/a/2057716

b) Trata de obtener siempre el objetivo `ALL` del fichero `Makefile`.

c) Trata de obtener siempre el objetivo `all` del fichero `Makefile.`

d) Ninguna de las anterior es.

Indica a qué paradigma hace referencia la siguiente definición:: "Jerarquía de autoridad similar al CC. Útil en la producción de un software similar a uno existente."

a) Paradigma cerrado Lesson 1. Punto 5

b) Paradigma aleatorio

c) Paradigma abierto

d) Paradigma sincronizado

Indica a qué paradigma hace referencia la siguiente definición:: "El equipo se estructura de manera libre en función de la iniciativa individual de los miembros. Útil cuando se requiere innovación."

a) Paradigma cerrado

b) Paradigma aleatorio Lesson 1. Punto 5

c) Paradigma abierto

d) Paradigma sincronizado

Indica a qué paradigma hace referencia la siguiente definición:: "Debe haber muy buena comunicación.. Son adecuados para resolver problemas complejos, pero pueden no ser tan eficientes como otros equipos."

a) Paradigma cerrado

b) Paradigma aleatorio

c) Paradigma abierto Lesson 1. Punto 5

d) Paradigma sincronizado

Indica a qué paradigma hace referencia la siguiente definición: "Las partes del problema nos sirven para organizar los miembros del equipo, los cuales suelen trabajar en estas partes del problema destacando la comunicación entre ellos."

a) Paradigma cerrado

b) Paradigma aleatorio

c) Paradigma abierto

d) Paradigma sincronizado Lesson 1. Punto 5

El dictador benevolente de por vida...:

a) Es importante que sea el originador del proyecto software. Lesson 1. Punto 7

b) Es imprescindible que sea el originador del proyecto software.

c) No son los encargados de velar por el proyecto que tutelan, simplemente cobran los beneficios del mismo.

d) Ninguna de las anteriores.

La rama `master` **o** `trunk` **es** `cutting Edge` **significa...:**

a) Los cambios no se pueden guardar.

b) No es necesario una rama estable con este tipo de rama.

c) Tiene las últimas modificaciones de nuestro software. Lesson 2. Punto 5

d) Ninguna de las anteriores.

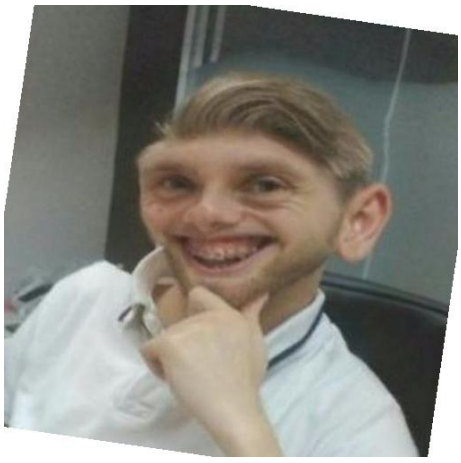
Indica cual es un a característica de `ccache` **:**

a) Mantiene estadísticas de aciertos/fallos.

b) Gestión automática del tamaño de la cache.

c) Puede cachear compilaciones con warnings.

d) Todas son correctas. Lesson 5. Punto 10



Preguntas enero DCA

Resueltas con su referencia/justificación

1. Para comenzar a usar autoconf es:
    a. Aconsejable usar autoscan
    b. Imprescindible usar autoscan
    c. Innecesario usar autoscan
    d. Obligatorio usar autoreconf

Aunque en la teoría se mencione a autoscan en una sección llamada “Imprescindibles” (tema 7 punto 21), no creemos que sea necesario realizar ese comando ya que puedes construir el configure.ac tú a mano desde cero (incluso usando ese comando deberás hacer modificaciones).

2. En la programación por contrato las precondiciones:
    a. se aplican solo a métodos de una clase
    b. se aplican solo a funciones independientes de cualquier clase
    c. se aplican tanto a métodos de una clase como a funciones independientes de cualquier clase.
    d. se aplican en el constructor de una clase

Tema 8, punto 3: “Un método o función puede tener varias precondiciones.”

3. En un lenguaje con soporte para la programación por contrato, las precondiciones y postcondiciones:
    a. se invocan cuando el programador lo indica en métodos de una clase
    b. se invocan primero las postcondiciones, luego las precondiciones y finalmente el código del método.
    c. se aplican en el constructor de una clase
    d. se invocan primero las precondiciones, luego el código del método y al final las postcondiciones.

Tema 8, puntos 3 y 4: “... para los clientes del método sus precondiciones son una obligación, mientras que para el método llamado representan lo que éste espera de los clientes que lo llaman.” y “Es este método el que debe asegurar que al terminar... todas sus postcondiciones se cumplen.”

4. Git es un SCVs:
    a. Evolución de BitKeeper
    b. Evolución de Mercurial
    c. Evolución de Monotone
    d. Influenciado por BitKeeper

Está influenciado por BitKeeper porque era sobre lo que se trabajaba antes para desarrollar Linux (Tema 10 punto 13). No es evolución de nada porque se comienza a desarrollar desde cero.

5. …
    a. trabajar con un Makefile por cada directorio de nuestro proyecto
    b. Trabajar con un único Makefile en la raíz de nuestro proyecto
    c. a que los archivos pasados a la orden make llamen a Makefile
    d. a crear como minimo un fichero Makefile.am si usamos Automake

6. Las autotools se componen de:
    a. Solo Autoconf
    b. Solo Automake
    c. Solo Autoconf y Automake
    d. Ninguna de las anteriores

Tema 6 punto 3: “AutoTools, las cuales constan de tres componentes: Autoconf, Automake, Libtool.”

7. La herramienta cmake:
    a. Permite tener solo un fichero CMakeList.txt en el directorio raíz del proyecto y ninguno en subdirectorios del mismo.
    b. No permite tener un fichero CMakeList.txt en cada subdirectorio del proyecto.
    c. Nos permite tener un fichero CMakeList.txt en cada subdirectorio del proyecto.
    d. Nos obliga a tener un fichero CMakeList.txt en cada subdirectorio del proyecto.

Tema 6 punto 29: “Solemos tener uno en el directorio principal de nuestro proyecto y uno por cada subdirectorio del mismo donde haya que construir algo.”

8. Gettext
    a. Traduce automáticamente todas las cadenas de nuestra aplicación.
    b. Necesita que las cadenas estén marcadas de forma distinta según el lenguaje al que se traducirán.
    c. Lleva a cabo la traducción palabra a palabra de una frase
    d. Solo traduce las cadenas marcadas de algún modo en nuestra aplicación

En teoría enseñan a que la traducción se hace marcando las cadenas en el Tema 9 punto 12, pero en el punto 14 pone que si le pasamos el argumento -a a xgettext extrae todas las cadenas aunque no estén marcadas. Esta no es segura.

9. La configuración de un proyecto software se hace:
    a. En función de los resultados tras una primera compilación.
    b. De forma distinta para cada máquina donde vayamos a compilarlo.
    c. Realizando unos test y chequeos antes de proceder a la compilación
    d. De forma manual para cada máquina donde lo vayamos a compilar

Tema 6 punto 8: “configure realiza las comprobaciones pedidas en configure.ac y finalmente…”

10. Un proyecto configurado con Autotools
    a. Solo se puede compilar en una máquina que tenga instalada el compilador con el que se configuró
    b. Necesita que esté instalado autoconf en la maquina donde se va compilar
    c. No se necesita que esté instalado autoconf en la máquina donde se va a compilar.
    d. Debe compilarse en la misma máquina donde se ha configurado.

Tema 6 punto 5: “Los guiones que produce no requieren tener instalado autoconf cuando son ejecutados. Son independientes de autoconf.”

11. Unas herramientas como Autotools

a. Solo se pueden emplear en proyectos en lenguaje C
b. Solo se pueden emplear en proyectos en lenguaje C++
c. Solo se pueden emplear en proyectos en lenguaje ObjectiveC
d. Ninguna de las anteriores

En ningún momento dice que esté restringido a un lenguaje en concreto. Además, aquí tienes un ejemplo hecho para Python: https://github.com/kostaz/python-autotools-tutorial

12. Un proyecto configurado con autotools puede constar de:
    a. Un único archivo llamado Makefile.am y varios config.h.in.
    b. Un archivo llamado autoconf.am
    c. Un archivo llamado configure.ac y varios Makefile.am
    d. Un archivo llamado Makefile.am y otro configure.ac

Como en la pregunta dice “puede”, ser tanto la c) como la d) en función de si el automake es recursivo o no (Tema 6 punto 23). Esta no es segura.

13. Lo que conocemos como Cmake:
    a. Se compone de varias herramientas
    b. Está formado solo por la aplicación cmake
    c. Está formado por las aplicaciones cmake y ctest
    d. Está formado por las aplicaciones cdash y cmake

Cuando instalas cmake en Ubuntu, vienen incluidas las herramientas CPack y CTest (aunque no CDash) que incluso puedes utilizar sin uso de Cmake (ejemplo CPack). Además, en Wikipedia se habla de “Familia de herramientas” (2º párrafo).

14. En el paso de test, el concepto Branch Coverage
    a. Se refiere a la cantidad de ramas empleadas en el controlador de versiones usado.
    b. Se refiere si estamos testeando la rama master
    c. Se refiere la cantidad de ramas del código empleadas
    d. Se refiere si estamos testeando la rama trunk

Branch Coverage es un concepto de cobertura de código (Tema 7 punto 10). El resto de opciones quedan descartadas porque hablan de sistemas de control de versiones.

15. En los Test de datos:
    a. Usamos datos reales y sintéticos
    b. Usamos solo datos sintéticos que hacen falta
    c. Usamos solo datos reales que son los que verdaderamente empleará la aplicación
    d. Nos dedicamos a comprobar los datos con los que trabaja la aplicación.

Tema 7 punto 8: “Tests de datos - Datos reales y sintéticos... ¿alguien dijo 30 de febrero?”

16. El paso de tests…
    a. sirve para mostrar los fallos de nuestro proyecto
    b. sirve para mostrar la ausencia de fallos de nuestro proyecto
    c. Sirve para mostrarnos las partes de nuestro poryecto que deben ser mejoradas
    d. no sirven para nada

Tema 7 punto 3: “El propósito del paso de tests es demostrar que en nuestro software existen fallos, no que está libre de ellos.”

17. En el GlassBox Testing:
    a. Tenemos acceso a parte del código a testear
    b. Tenemos acceso sólo al código del programa principal a testear
    c. Tenemos acceso a todo el código a testear
    d. Tenemos acceso a un módulo del código a testear

Si no tuviéramos acceso al código a probar, no estaríamos haciendo el WhiteBox testing propiamente dicho (testeas el código baśandote en las estructuras de control). Tema 7 punto 6: “Al tener acceso al código debemos fijarnos al construir los tests en las sentencias if-then-else, bucles, try-catch presentes en el código a testear”