
**Universidad de Costa Rica**
**Escuela de Ciencias de la Computación e Informática**
**Semestre I - 2021**
**Curso CI-0113 - Programación II**
**Grupo 03 y Grupo 06**
**Profesor:** Edgar Casasola Murillo
**Fecha:** 28 de julio 2021
**Hora:** 11 a.m. a 3:00 p.m.

# EXAMEN II

**NO SE TOMARÁ EN CUENTA SU TRABAJO SI "COPIA" CÓDIGO FUENTE DE TERCEROS, EL CÓDIGO DEBERÁ SER ESCRITO POR USTEDES CON LO VISTO EN EL CURSO. SI PUEDEN HACER USO DEL CÓDIGO VISTO EN CLASE. ADEMÁS, PUEDEN HACER USO DE LAS CLASES DE LA BIBLIOTECA ESTÁNDAR DE PLANTILLAS STL EN LOS EJERCICIOS QUE ASÍ LO INDIQUEN.**

## Ejercicio #1 Algoritmo de inserción en un Árbol Rojo Negro ( 30 % )

Con el algoritmo visto en clase dibuje las secuencia de modificaciones ( una imagen para cada inserción como mínimo ) al insertar una secuencia de valores en un árbol Rojo Negro. Pueden haber varias imágenes para cada inserción en caso de ser necesario. En cada paso debe indicar las operaciones que aplicó usando las abreviaturas vistas en clase, a saber: CCR, CF, RSI, RSD, RDI, RDD, RC. Cada estudiante cuenta con una secuencia personal asignada de números en el archivo README.MD del repositorio que se construyó para tal efecto en git.ucr.ac.cr/examen2. Hay un repositorio particular para cada estudiante. En el README.MD se indica el Número de Estudiante eXX asignado y luego la línea con los valores que debe insertar siguiendo el orden de izquierda a derecha.

Si hay n números para ser insertados habrá como mínimo n estados del árbol que deben mostrar, idealmente más, ya que en algunos casos se llevarán a cabo varias modificaciones por inserción. Pueden trabajar en alguna herramienta para hacer diagramas tipo Paint y entregarlo en formato pdf o en una hoja de papel y adjuntar la foto posteriormente (asegurarse que es legible lo que hizo). Además si hay una secuencia de fotos deben ir numeradas en el orden de creación.

## Ejercicio #2 Uso de la STL ( 40% )

El objetivo de este ejercicio es que demuestren su manejo de la biblioteca STL, pueden usar cualquier contenedor y algoritmo que consideren aptos.

Deberán construir un programa base para un Servicio de creación y distribución de fichas para un Juego de Dominó. El programa deberá tener las siguientes funciones:

1. Crear todas las fichas para un juego de dominó con valores de 0 a 6. En este caso se construirá desde la ficha [0 : 0] hasta la ficha [6 : 6]. No pueden haber fichas repetidas. La ficha [X : Y] es lo mismo que la ficha [Y : X], para cada par de valores X Y. En otras palabras, el orden de los dos valores no importa, simplemente significa que está viendo la misma ficha pero al revés.

- Su programa una vez creadas las fichas las debe desordenar en forma aleatoria (random shuffle) antes de poder repartirlas.
- Seguidamente deberá construir 3 grupos de 7 fichas sacadas de las que están revueltas. Un grupo para cada uno de los 3 jugadores. Cada jugador será un número de 0 a 2. No debe construir ninguna clase Jugador. Cada grupo será para darle esas fichas al jugador que las solicite. Asuma que cuando un jugador pide ver sus fichas en pantalla los otros no pueden ver esa misma pantalla, así que cada quien tiene fichas secretas que los demás no pudieron ver.
- Las fichas sobrantes quedarán en una Cola de Fichas Sobrantes de la cual se tomará una ficha cada vez que un jugador la pida.

  Escriba un main que lleve a cabo el proceso de creación del mazo de fichas, desordena, crea los 3 grupos para cada jugador y la cola de fichas desordenadas.

- Para verificación deberá Imprimir en pantalla las fichas antes y después de ser desordenadas.
- Posteriormente, el programa permite leer el número de un jugador y le muestra sus fichas asignadas.
- Finalmente extrae todas las fichas sobrantes de la Cola de Fichas Sobrantes hasta que no queden más fichas, mostrándolas una a una en pantalla.

**Puede crear cualquier clase extra o método que consideren necesarios. Recuerde que debe usar contenedores de la stl, algoritmos e iteradores para la solución.**

**Ejercicio #3 Poliformismo y plantillas ( 30% )**

Construya una jerarquía de clases y una plantilla de clase para que el main adjunto funcione correctamente.

```
#include "Fabrica.h"
```

```
#include "Transformador.h"
#include "Aumentador.h"
#include "Reductor.h"
#include "Invertidor.h"
#include <string>
#include <iostream>
using namespace std;

// Salida esperada:
// Entra: AbCd Sale: ABCD
// Entra: AbCd Sale: dCbA
// Entra: AbCd Sale: abcd

int main(){
  string original = "AbCd";
  Transformador * transformador[3];
  Fabrica<Aumentador> f1;
  Fabrica<Invertidor> f2;
  Fabrica<Reductor> f3;
  transformador[0]= f1.producir();
  transformador[1]= f2.producir();
  transformador[2]= f3.producir();
  for( int i=0; i < 3; ++i){
    cout << "Entra: " << original << " ";
    cout << "Sale: " << transformador[i]->transformar(original) << endl;
    delete transformador[i];
  }
  return 0;
}
```

La jerarquía debe ofrecer una solución polimórfica para crear Transformadores. Deberán existir tres tipos de Transformador llamados: Aumentador, Reductor e Invertidor. Dado que todas las Fabricas son iguales y lo que hacen es producir() un producto específico (en este caso el producto son instancias de las clases derivadas de Transformador), note en el código del main que para crear las Fabricas específicas de cada subclase de Transformador se deberá construir una Plantilla o Template que para cada tipo de Transformador genera una Fabrica específica que produce ese tipo de Transformador.

Deberá construir los .h y .cpp necesarios para que el main.cpp funcione así como está.
Para la jerarquía deberá crear:

Clase Transformador

Descripción: Un Transformador es una clase que tiene un método transformar que recibe un string y lo transforma a otro string.

- Clase abstracta
- Debe tener un método llamado: string transformar(string): recibe un string y retorna una **copia** del string modificado según la función de la clase Transformador a la que pertenezca.

Clase Aumentador

- Es una subclase de la clase Transformador.
- Implementa el método transformar que recibe un string y retorna una copia del string transformado a mayúscula.
- Ejemplo, si se tiene:
  Aumentador aumentador;
  Entonces: aumentador.transformar(“hola”) retorna “HOLA”

Clase Reductor

- Es una subclase de la clase Transformador.
- Implementa el método transformar que recibe un string y retorna una copia del string transformado a minúscula.
- Ejemplo, si se tiene:
  Reductor reductor;
  reductor.transformar(“HOLA”) retorna “hola”

Clase Invertidor

- Es una subclase de la clase Transformador.
- Implementa el método transformar que recibe un string y retorna una copia del string transformado a un string con invertido con las mismas letras.
- Ejemplo, si se tiene:
  Invertidor invertidor;
  invertidor.transformar(“hola”) retorna “aloh”

Cree una plantilla de clase llamada Fabrica< T > que construya clases emplantilladas con Fábricas para cada tipo de Transformador.

Compile el main.cpp con sus clases para probar su correcto funcionamiento.

**FORMA DE ENTREGA:**

**No se puede, ni se deberá subir a git** su solución, sino que **se entregará en** el enlace para entrega de solución de examen II en **mediación.virtual.ucr.ac.cr.**

**Debe subir su solución al enlace respectivo en mediacionvirtual.ucr.ac.cr antes de las 3:00 p.m. del día Miércoles 28 de julio de 2021.**

Su solución puede consistir de:

1. Si hoy no cuenta con computador adecuado para la solución del examen o si se va la electricidad o internet (Casos especiales conocidos):
   Carpeta con Fotos o Documento en formato .pdf de su solución a puño y letra en caso de que no cuente con computador para trabajar el día de hoy. Explique su solución y escriba todo el código fuente .h y .cpp de cada respuesta.
2. Si no surge ningún inconveniente, lo que se espera es que usted suba una carpeta con subcarpetas (una para cada pregunta) debidamente identificada con su nombre. **Debe incluir todo el código .h y .cpp y main de prueba con el que usted verificó el funcionamiento de las respuestas a cada pregunta.**

   Puede replicar código en las carpetas con tal de que sea compilable sin problema. Incluya un archivo de texto llamado README.md donde explique como compilar, ejecutar cada solución y posibles problemas encontrados en caso de que no le compile.

**PUNTOS EXTRA**

Como parte de este examen y en ambos casos se dará un 10% extra sobre la nota del examen por entregar una solución extra debidamente documentada y funcional del examen. Cuya entrega será el **Viernes 30 de julio a las 11:59 pm en el enlace indicado en mediación virtual.**