PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA

Segundo Examen

(Primer Semestre 2023)

Duración: 2h 50 min.

- En cada función el alumno deberá incluir, a modo de comentario, la estrategia o forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- No puede emplear STL, Plantillas o funciones no vistas en los cursos de la especialidad.
- Los programas deben ser desarrollados en el lenguaje C++. Si la implementación es diferente a la estrategia indicada o no la incluye, la pregunta no será corregida.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 50% del puntaje asignado a dicha pregunta.
- Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado. El orden será parte de la evaluación.
- Se utilizarán herramientas para la detección de plagios, por tal motivo si se encuentran soluciones similares, se anulará la evaluación a todos los implicados y se procederá con las medidas disciplinarias dispuestas por la FCI.
- **Solo está permitido acceder a la plataforma de PAIDEIA, cualquier tipo de navegación, búsqueda o uso de herramientas de comunicación se considera plagio por tal motivo se anulará la evaluación y se procederá con las medidas disciplinarias dispuestas por la FCI.**
- **Los programas deben ser desarrollados utilizando nombres para las funciones y variables en español, al igual que los comentarios. El uso de otro idioma anula su respuesta.**
- Para esta evaluación solo se permite el uso de las librerías **iostream, iomanip, climits cmath, fstream y cstring**
- Su trabajo deberá ser subido a PAIDEIA.
- Los archivos deben llevar como nombre su código de la siguiente forma codigo_EX2_P# (donde # representa el número de la pregunta a resolver).

Resuelva solo 2 de las siguientes preguntas:

Pregunta 1 (10 puntos)

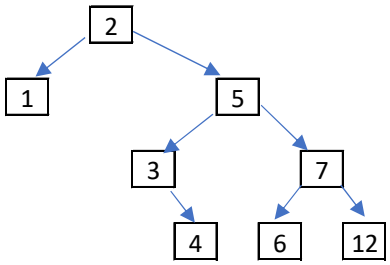
El uso de tokens permite reforzar la seguridad en la etapa de autenticación de un usuario en un sistema, ya que permite la autenticación de dos factores (2FA). Se le solicita primero al usuario sus credenciales (usuario y contraseña) como primer factor de autenticación. Luego se le solicita al usuario que ingrese un grupo de números generados por el token físico.

La simulación de la verificación del segundo factor (códigos generados por el token físico) funciona de la siguiente manera:

1. El usuario ingresa un grupo de 8 códigos (números enteros de 1 al 9), los cuáles pueden ser repetidos.
   Ejemplo: 2, 5, 2, 1, 5, 6, 3, 4

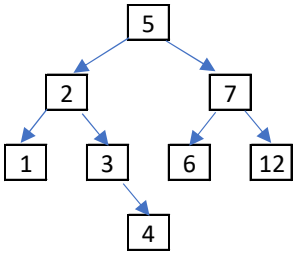
2. Los códigos se insertan en un árbol de búsqueda binario. La función debe realizar la inserción en el ABB de manera recursiva. Además, **debe verificar si ya existe el código en el árbol,** si eso ocurre se debe generar un nuevo código que resulta de la suma del código que se desea insertar con el nodo mayor en el momento de la inserción.

Ejemplo:



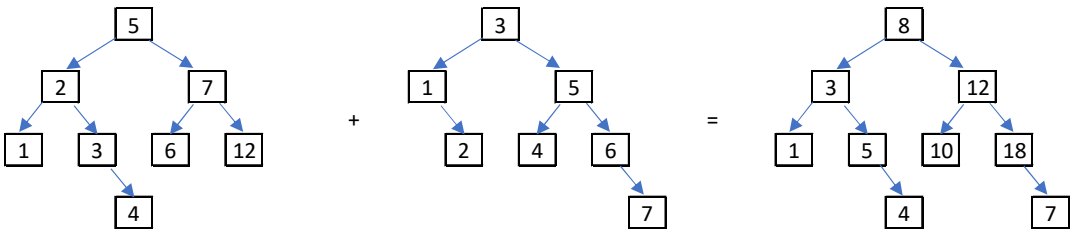
3. Luego el árbol de búsqueda binario generado, debe pasar por un proceso de balanceo.

Ejemplo:

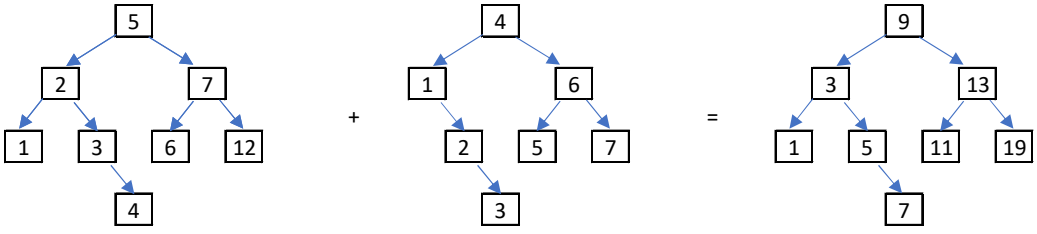


4. El ABB balanceado es "sumado" con otro ABB generado internamente por el sistema para verificar los códigos generados por el token físico. Si la "suma" de ambos árboles genera un árbol binario de búsqueda, el sistema le permite el acceso al usuario. Puede usar un árbol auxiliar para generar el árbol suma.
NOTA: La "suma" de los árboles se hace nodo a nodo según el nivel en el que se encuentra:

FALSO



VERDADERO



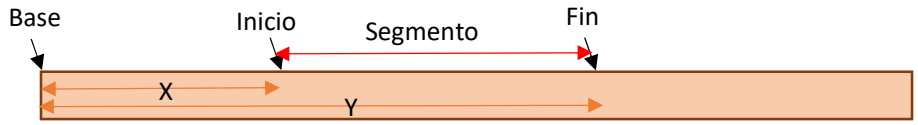
Se le solicita:

a) Establecer la función principal, las estructuras de datos y demás funciones que permitan manejar lo descrito sobre la simulación de verificación del segundo factor. Recuerde que al final, luego del desarrollo de las funciones solicitadas, el usuario debe de recibir un mensaje indicando "Acceso otorgado" o "Acceso denegado" según sea el caso. (1 punto)
b) Elaborar la función **inserta_nodo**() que reciba como parámetros la raíz del árbol y un código generado por el token físico, permitiendo insertar, usando recursividad, una hoja en el ABB y considerando lo descrito en el punto 2. (3 puntos)
c) Elaborar la función **balancea_arbol**() que, usando recursividad, permita cumplir con el paso 3. (3 puntos)
d) Elaborar la función **suma_arboles**() que permita cumplir con el paso 4. La función recibe como parámetros dos ABB y devuelve verdadero (1) si la "suma" de los ABB resulta un árbol binario de búsqueda, o falso (0) en el caso contrario. (3 puntos)

NOTA: En el módulo principal, se debe invocar a cada una de las funciones que se le solicita desarrollar en b, c y d, debe de mostrar (imprimir/visualizar en preorden) el árbol con el que se ha trabajado en dicha función.

Pregunta 2 (10 puntos)

Una empresa maderera ha decidido comprar un robot aserrador, el mismo se encarga de cortar **N** troncos de madera de acuerdo con las **M** presentaciones que maneja la empresa. Se sabe que cada presentación, está determinada por el punto de inicio del corte a **X** metros con respecto a la base, un punto final a **Y** metros con respecto a la base y una ganancia o beneficio para la empresa, ya que cada segmento del tronco tiene diferente calidad y uso, como pueden ser muebles, adornos, soportes, etc. Para optimizar esta tarea se necesita que el robot seleccione las presentaciones de tal forma que maximice el beneficio o ganancia, sin importar el desperdicio. **Solo debe indicar la ganancia máxima que puede obtener seleccionando las presentaciones adecuadas**. Desde luego los residuos no pueden ser unidos o pegados para formar una presentación, solo se desechan. Por lógica no se pueden seleccionar dos presentaciones que se intersecan dentro del tronco.



Ejemplo 1:

Si la empresa tiene: N = 10 troncos de madera y M = 4 presentaciones.

	Presentación 1	**Presentación 2**	**Presentación 3**	**Presentación 4**
Inicio (m)	5	2	6	4
Fin (m)	10	4	12	15
Beneficio (S/.)	30	40	80	100

La respuesta es que la ganancia máxima que se puede obtener es igual a S/. 1400. Ya que se elegirían las presentaciones 2 y 4 con una ganancia de S/. 140 por cada tronco cortado. Esta solución es adecuada ya que las dos presentaciones no se intersecan entre ellas. No se puede elegir la presentación 1 o 3 adicionalmente porque empiezan dentro de los segmentos 2 y 4 que brindan la mayor ganancia. Por ejemplo, la 1 se interseca con la 4. En el caso de la 3 se interseca con la 4.

a) Utilizando **programación dinámica,** desarrolle un programa que muestre la ganancia máxima que se puede obtener con N troncos y M presentaciones. Para esta pregunta debe usar los datos que se muestran en el ejemplo, **no** deben ingresarlos por el teclado o archivo. No puede usar registros ni TADs, solo arreglos y matrices de enteros (5.0 puntos)

Un criador de aves ornamentales dedicado a la crianza de canarios y agapornis ha decido instalar las jaulas para sus pequeñas mascotas. Las jaulas se instalarán en 2 filas una al lado de otra, el problema es que no se pueden colocar 2 agapornis uno al lado del otro de forma horizontal o vertical ya que esta especie es muy agresiva y territorial eliminando a sus iguales. Pero si tolera a aves pequeñas como los canarios. Por tal motivo se necesita saber cuántas combinaciones se pueden construir con **n** columnas.

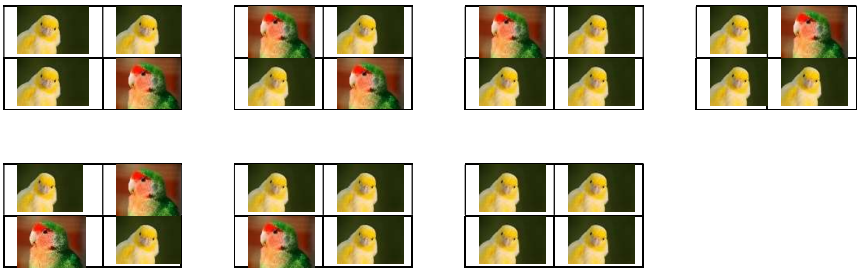
Ejemplo 1:

Si n=1 se podrán formar las siguientes combinaciones:

Por tal motivo con n=1 columnas se podrán formar 3 combinaciones diferentes.

Ejemplo 2:

Si n=2 se podrán formar las siguientes combinaciones:



Por tal motivo con n=2 columnas se podrán formar 7 combinaciones diferentes.

Se solicita:

b) Utilizando **programación dinámica,** desarrolle un programa que calcule la cantidad de combinaciones que se pueden formar con **n** columnas. Para esta pregunta debe usar los datos que se muestran en el ejemplo, **no** deben ingresarlos por el teclado o archivo. No puede usar registros ni TADs, solo arreglos y matrices de enteros (5.0 puntos)

Para ambas preguntas debe mostrar el arreglo o matriz de soluciones propia de la programación dinámica, en caso contrario su solución no es válida. Recuerde que ambas preguntas deben ser totalmente iterativas. Además, debe presentarse en dos proyectos o archivos .cpp distintos de acuerdo con la opción desarrollada.

Pregunta 3 (10 puntos)

Durante todo el semestre, Bowser ha librado muchas batallas contra el reino champiñón y aliados, sin embargo, la incursión final por la toma del reino champiñon ha llegado. Para esta incursión, Bowser prepara lo mejor de su ejército y está convocando a 3 tipos de guerreros: Aire, Tierra y Agua.

Como se trata de la incursión final, Bowser ha preparado un ejército con un gran número de guerreros por cada tipo, esto debido a que tiene que sortear "X" frentes de defensa que ha preparado el reino champiñón para detener el ataque del ejército de Bowser antes de llegar a la princesa Peach. Cada frente de defensa del reino champiñón está formado exactamente por "Y" ejércitos, donde cada ejército tiene una cantidad variable de guerreros por cada tipo (Airea, Tierra y Agua), donde la cantidad de guerreros por cada tipo siempre será un número mayor o igual a 10 y menor o igual a 35.

Para que el ejército de Bowser pueda sortear un frente de ataque, debe elegir a uno de los ejércitos del reino champiñón de dicho frente y vencerlo. En caso el ejército de Bowser pierda en alguno de los frentes, automáticamente se debe indicar que "El Reino Champiñon ha ganado". Si el ejército de Bowser logra sobrevivir a todos los frentes y llegar a la reina Peach se debe indicar que "El Reino Champiñon ha caído".

Para simular la pelea entre el ejército de Bowser y un ejército del reino Champiñon de un frente de ataque, debe considerar lo siguiente:

- 1 guerrero del tipo Agua del reino champiñón se elimina con un guerrero del tipo Agua del ejército de Bowser.
- 2 guerreros del tipo Agua del reino champiñón se eliminan con un guerrero del tipo Tierra del ejército de Bowser.
- 3 guerreros del tipo Agua del reino champiñón se eliminan con un guerrero del tipo Aire del ejército de Bowser.
- 1 guerrero del tipo Tierra del reino champiñón se elimina con 2 guerreros del tipo Agua del ejército de Bowser.
- 1 guerrero del tipo Tierra del reino champiñón se elimina con 1 guerrero del tipo Tierra del ejército de Bowser.
- 2 guerreros del tipo Tierra del reino champiñón se eliminan con 1 guerreros del tipo Aire del ejército de Bowser.
- 1 guerrero del tipo Aire del reino champiñón se elimina con 3 guerreros del tipo Agua del ejército de Bowser.
- 1 guerrero del tipo Aire del reino champiñón se elimina con 2 guerrero del tipo Tierra del ejército de Bowser.
- 1 guerrero del tipo Aire del reino champiñón se elimina con 1 guerrero del tipo Aire del ejército de Bowser.

Considere que siempre el ejército del frente del reino champiñón va a atacar al ejército de Bowser, siguiendo lo indicado en el punto anterior. El ataque entre guerreros se debe desarrollar en la precedencia de primero eliminar todos los guerreros agua del ejército de Bowser, luego los del tipo Tierra y por último los del tipo Aire. Si alguno de los ejércitos (champiñón o Bowser) se queda sin ningún guerrero (de ningún tipo) entonces ese ejército perdió la batalla.

A continuación, se presenta un ejemplo de ejecución de una incursión final:

Ingreso:

Ingrese la cantidad de guerreros del tipo Agua del ejército de Bowser: 150

Ingrese la cantidad de guerreros del tipo Tierra del ejército de Bowser: 80

Ingrese la cantidad de guerreros del tipo Aire del ejército de Bowser: 40

Ingrese el número de frentes del reino champiñón: 2

Ingrese el número de ejércitos del reino champiñón por cada frente: 2

Ingrese la cantidad de guerreros del tipo Agua del ejército 1 del Frente 1 del reino champiñón: 30

Ingrese la cantidad de guerreros del tipo Tierra del ejército 1 del Frente 1 del reino champiñón: 20

Ingrese la cantidad de guerreros del tipo Aire del ejército 1 del Frente 1 del reino champiñón: 10

Ingrese la cantidad de guerreros del tipo Agua del ejército 2 del Frente 1 del reino champiñón: 35

Ingrese la cantidad de guerreros del tipo Tierra del ejército 2 del Frente 1 del reino champiñón: 25

Ingrese la cantidad de guerreros del tipo Aire del ejército 2 del Frente 1 del reino champiñón: 15

Ingrese la cantidad de guerreros del tipo Agua del ejército 1 del Frente 2 del reino champiñón: 18

Ingrese la cantidad de guerreros del tipo Tierra del ejército 1 del Frente 2 del reino champiñón: 15

Ingrese la cantidad de guerreros del tipo Aire del ejército 1 del Frente 2 del reino champiñón: 12

Ingrese la cantidad de guerreros del tipo Agua del ejército 2 del Frente 2 del reino champiñón: 22

Ingrese la cantidad de guerreros del tipo Tierra del ejército 2 del Frente 2 del reino champiñón: 13

Ingrese la cantidad de guerreros del tipo Aire del ejército 2 del Frente 2 del reino champiñón: 10

Salida:

El reino champiñón ha caído, el ejército de Bowser venció al ejército 1 del frente 1 y ejército 2 del frente 2 y le quedaron 0 guerreros de tipo Agua, 56 guerreros del tipo Tierra y 40 guerreros del tipo Aire.

Para el caso de prueba presentado se tiene la siguiente Figura 1, donde la línea roja representa la solución encontrada en la salida.

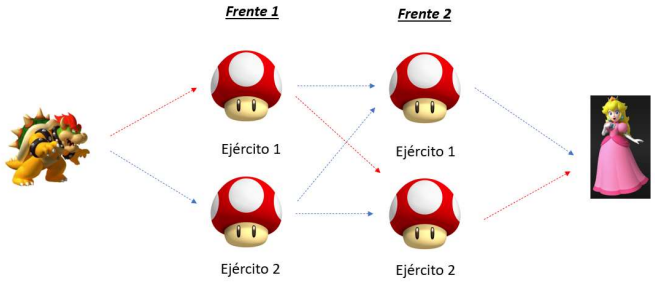


Figura 1. Ejemplo del caso de prueba.

Se le pide:

a) Defina las estructuras y desarrolle las funciones necesarias para soportar el ingreso de datos descrito en el problema. Recuerde que la cantidad de frentes y cantidad de ejércitos por cada frente es variable. (2 puntos).
b) Desarrolle una función utilizando la estrategia de Backtracking que permita calcular una de las rutas que debe seguir el ejército de Bowser para llegar a la reina Peach e indique cuantos guerreros le quedarían al ejército de Bowser por cada tipo. (6 puntos).
c) Realice las modificaciones necesarias a la función anterior para que permita calcular todas las rutas posibles que debe seguir el ejército de Bowser para llegar a la reina Peach e indique cuantos guerreros le quedarían al ejército de Bowser por cada tipo por cada ruta. (2 puntos).

Profesores del curso:

David Allasi
Fernando Huamán
Rony Cueva

San Miguel, 15 de julio del 2023