**Universidad Interamericana de Panamá Facultad de Ingeniería, arquitectura y diseño Escuela de Ingeniería y Sistemas.**

Materia: Estructura de Datos II Cuatrimestre: IIQ\_2019 Código: 301-00040 Facilitador: Leonardo Esqueda.

Nombre completo: \_\_EDGAR FERNANDO ROJAS CARRANZA\_\_ ID: \_\_ao-8924-08\_\_

Carrera: \_\_INGENIERIA INDUSTRIAL Y DE SISTEMAS\_\_\_

**PARCIAL #1**

**Instrucciones: La prueba parcial está enfocada en dos (2) pates: Teórico y Práctico; constando con 13 problemas. Lea detenidamente antes de responder.**

**A) Teoría:**

1. ¿Qué son las Estructuras de Datos? 5ptos

2. ¿De dónde vienen las estructuras de datos? 5ptos

3. ¿Mencione las dos (2) ramas de la estructura de datos interna? 5ptos

4. ¿Cómo interactúan las estructuras de datos internas y externas? 5ptos

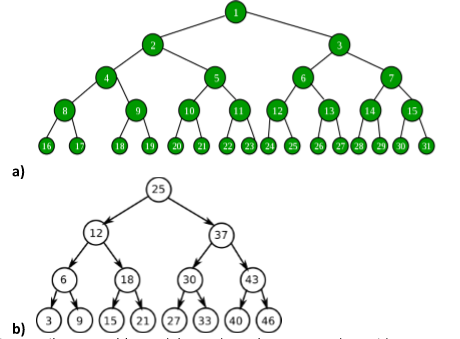
5. ¿Cómo funcionan los árboles Binarios? 5ptos

6. Describa un caso en donde usted podría realizar y utilizar un árbol binario. 5ptos

7. Mencione y explique 5 propiedades de los métodos de ordenamiento de los arboles binarios. 5ptos

8. Distribuya los nodos -5,2,-11,4,-13,5,3,-14,1,6,10,-12,8 en un árbol binario y determine ¿Cuáles son los nodos hoja? ¿Cuáles son nodos de 2 hijos? 5ptos

9. Realice el recorrido pre orden, inorden y postorden de los árboles siguientes: 5ptos



**B) Práctica:**

1. Realice un algoritmo lógico el cual explique y demuestre el proceso de inserción de los elementos del problema teórico número dos (2) en un árbol binario. 15ptos.

2. Desarrolle un programa en el lenguaje C++ o Python en donde cargue por defecto los elementos del árbol: 9 – A y realice los recorridos: preorden, inorden y postorden 15ptos

3. Las Torres de Hanoi es un juego matemático que consiste en tres varillas verticales y un número indeterminado de discos que determinarán la complejidad de la solución. No hay dos discos iguales, están colocados de mayor a menor en una varilla ascendentemente, y no se puede colocar ningún disco mayor sobre uno menor a él en ningún momento. El juego consiste en pasar todos los discos a otra varilla colocados de mayor a menor ascendentemente. Leyenda: Dios al crear el mundo, colocó tres varillas de diamante con 64 discos en la primera. También creó un monasterio con monjes, los cuales tienen la tarea de resolver esta Torre de Hanoi divina. El día que estos monjes consigan terminar el juego, el mundo acabará. El mínimo número de movimientos que se necesita para resolver este problema es de 264-1. Si los monjes hicieran un movimiento por segundo, los 64 discos estarían en la tercera varilla en poco menos de 585 mil millones de años. Como comparación para ver la magnitud de esta cifra, la Tierra tiene como 5 mil millones de años, y el Universo entre 15 y 20 mil millones de años de antigüedad, sólo una pequeña fracción de esa cifra.

Resolución: el problema de las Torres de Hanoi es curioso porque su solución es muy rápida de calcular, pero el número de pasos para resolverlo crece exponencialmente conforme aumenta el número de discos. Para obtener la solución más corta, es necesario mover el disco más pequeño en todos los pasos impares, mientras que en los pasos pares sólo existe un movimiento posible que no lo incluye. El problema se reduce a decidir en cada paso impar a cuál de las dos pilas posibles se desplazará el disco pequeño:

El algoritmo en cuestión depende del número de discos del problema. Si inicialmente se tiene un número impar de discos, el primer movimiento debe ser colocar el disco más pequeño en la pila destino, y en cada paso impar se le mueve a la siguiente pila a su izquierda (o a la pila destino, si está en la pila origen). La secuencia será DESTINO, AUXILIAR, ORIGEN, DESTINO, AUXILIAR, ORIGEN, etc. Si se tiene inicialmente un número par de discos, el primer movimiento debe ser colocar el disco más pequeño en la pila auxiliar, y en cada paso impar se le mueve a la siguiente pila a su derecha (o a la pila origen, si está en la pila destino). La secuencia será AUXILIAR, DESTINO, ORIGEN, AUXILIAR, DESTINO, ORIGEN, etc. 15ptos.

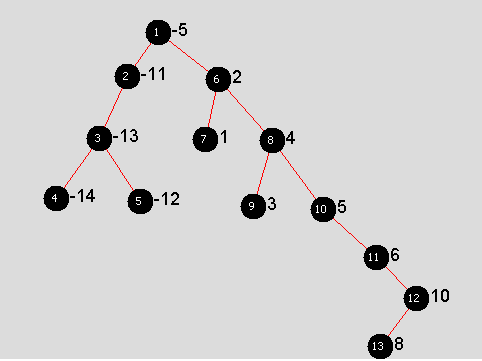
**Desarrollo.**

1. Cuando hablamos de programación, la estructura de datos está representada por una forma determinada que tenemos de organizar los datos de un equipo informático para que podamos utilizarlos de la manera más efectiva posible. Dependiendo del tipo de aplicación o recurso que vayamos a usar requeriremos una estructura de datos independiente y distinta a las demás, dado que cada una encaja en el contexto de forma determinada y con una serie de objetivos.

Con estas estructuras tenemos la posibilidad de administrar todo tipo de datos sin ningún tipo de obstáculo, algo que en la actualidad se usa en la red para poder llevar a cabo, por ejemplo, los sistemas de indexado de contenidos. Y también juegan un papel clave en la creación de los mejores algoritmos, así como en su uso con lenguajes de programación que se benefician de ellas.

Las estructuras de datos es una rama de las ciencias de la computación que estudia y aplica diferentes formas de organizar información dentro de una aplicación, para manipular, buscar e insertar estos datos de manera eficiente.

1. La estructura de datos viene de las actividades de la vida real.
2. Las dos ramas de la estructura de datos interna son estáticas y dinámica.
3. –
4. El árbol binario es el caso más simple de árbol de orden N, cuando N vale 2. Su especificación se puede hacer considerando un valor constante, el árbol nulo, y un constructor de árboles a partir de un elemento y dos árboles.
5. En un navegador de búsqueda.
6. –



**Nodos hojas:** -14, -12, 1, 3, 8.

**Nodos de 2 hojas:** -13, -5, 2, 4.

1. A.

**Preorden:** 1, 2, 4, 8, 16, 17, 9, 18, 14, 5, 10, 20, 21, 11, 22, 23, 36, 12, 24, 25, 13, 26, 27, 7, 14, 22, 29, 15, 30, 31.

**Posorden:** 16, 8, 17, 4, 18, 9, 19, 2, 20, 10, 21, 5, 22, 11, 23, 24, 12, 25, 6, 26, 13, 27, 3, 28, 14, 29, 7, 30, 15, 31, 1.

**Inorden:** 16, 8, 17, 4, 18, 9, 19, 2, 20, 21, 5, 22, 11, 23, 1, 24, 12, 25, 6, 26, 13, 24, 28, 14, 29, 7, 30, 15, 31.

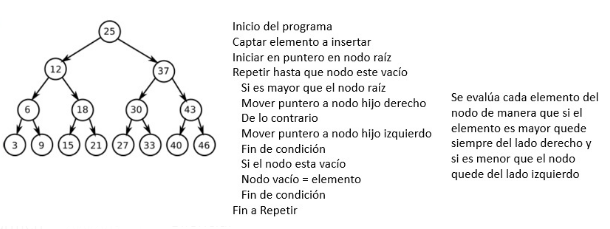
B.

**Preorden**: 25, 12, 6, 3, 9, 18, 15, 21, 37, 30, 27, 33, 43, 40, 46.

**Posorden**: 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 37, 40, 43, 46, 25.

**Inorden**: 3, 6, 9, 12, 15, 18, 21, 25, 27, 30, 33, 37, 40, 43, 46.

**PRACTICA:**



1. –
2. #Clase llamada nodo
3. class Nodo:
4. #Constructor con los argumentos
5. def init(self,numero, izq=None, der=None):
6. #Punteros
7. self.numero = numero
8. self.izq = izq
9. self.der = der
10. #
11. # #Retornar el valor
12. # def str(self):
13. # return self.poss
14. #Clase del arbol binario
15. class aBinarios:
16. #metodo constructor y el atributo raiz|
17. def init(self):
18. self.raiz = None
19. #creamos la funcion agregar
20. def agregar(self):
21. #Comparamos si la raiz esta vacia, de ser cierto añadiremos el nodo en la Raiz
22. padre = self.raiz
23. self.raiz = nodo(1)
24. For i=2 in Range(32):
25. if padre.izq = None:
26. padre.izq= nodo(i)
27. else:
28. if padre.der == None
29. padre.der= nodo(i)
30. else:
31. if (padre.izq != None) && (padre.izq == None)
32. padre = padre.izq
33. else:
34. if (padre.izq != None) && (padre.izq != None)
35. padre = padre.der
36. def inorder(self, a):
37. if a == None:
38. return None
39. else:
40. self.inorder(a.left)
41. print(a.dato)
42. self.inorder(a.right)
43. def preorder(self, a):
44. if a == None:
45. return None
46. else:
47. print(a.dato)
48. self.preorder(a.left)
49. self.preorder(a.right)
50. def postorder(self, a):
51. if a == None:
52. return None
53. else:
54. self.postorder(a.left)
55. self.postorder(a.right)
56. print(a.dato)
57. #se crea funcion que permite obtener la raiz
58. def getRaiz(self):
59. return self.raiz
60. #creamos un menu
61. if name == "main":
62. #Creamos una instancia de la Clase aBinarios
63. ab = aBinarios()
64. poss=0
65. while(True):
66. #opciones del menu
67. print("Arboles\_Binarios\n"+
68. "1. Agregar Usuario Nuevo\n"+
69. "2. Buscar Usuario\n"+
70. "3. Salir\n")
71. num = input("Ingrese la opcion: ")
72. if num == "1":
74. ab.agregar(nod)
75. elif num == "2":
76. nombre= input("Ingrese el nombre del Usuario: ")
77. print("El Usuario se encuentra en la posicion: ",ab.Buscar(nombre,ab.getRaiz()))
78. elif num == "3":
79. exit()