Titulación: GRADO INGENIERIA INFORMÁTICA
Asignatura: TÉCNICAS DE PROGRAMACIÓN
Fecha: 2022/06/04 – 9:00

UAX
UNIVERSIDAD ALFONSO X EL SABIO

Examen ordinario 2C

Nombre y apellidos:
NP:
Firma:

Firma vigilante:

Sede: .....**Modelo de Examen: Único**

Instrucciones del examen:

La duración del examen es de 1:45 min

Todas las preguntas deben responderse en archivos digitales. Cada una con su código y comentarios explicativos que el alumno considere oportunos.

En los casos que se indican nombres de proyectos, variables y mensajes específicos, se deben reproducir exactamente (respetando también mayúsculas y minúsculas) o se restarán puntos de la calificación final.

Al terminar salvar el proyecto y crear un archivo comprimido (.zip, .7z o .rar) de la carpeta para enviarlo.

Preguntas:

Ejercicio 1: (2,5 puntos).

Como paso previo a la reorganización del listín telefónico de cierta comunidad, se van a organizar en una tabla hash cerrada todos los números de teléfono atendiendo a una resolución cuadrática de colisiones. Suponga que la tabla es de tamaño 18 y que la función de hash es fHash= (k+3) %tamTabla, siendo k la terminación del número de teléfono. Las terminaciones de los números de teléfono son: 9332, 4569, 2231, 7810, 5554, 3852, 4041, 5699, 1209 y 6693.

Crear un archivo, Ejercicio1.txt y detalla lo siguiente:

a) Detalle el estado de la tabla hash si hace un rehash con factor de carga de 55%. Tenga en cuenta que al hacer el rehash el nuevo tamaño de la tabla es el doble menos 1.
b) Detalle los pasos de la inserción de las claves en una tabla abierta.

Ejercicio 2: (2,5 puntos).

Se desea un método que calcule la siguiente fórmula iterativa:

$$f(n) = \begin{cases} 1, n = 0 \\ n * f(n-1), 0 < n < 3 \\ 2n * f(n-2), 6 > n \geq 3 \\ 3n * f(n-3), 10 \geq n \geq 6 \\ 4n * f(n-4), n > 10 \end{cases}$$

Cree un archivo Ejercicio2.java y escriba método con programación dinámica para calcular el valor de $f(n)$.

Ejercicio 3: (2,5 puntos).

En un almacén se venden tubos utilizando largos estándar de 1m, 1'5m, 4m y 5m. Cuando un cliente realiza un pedido solicita un número dado de metros, por ejemplo 18m. El problema es calcular cuántos tubos y de qué tamaños enviar al cliente.

Cree un archivo Ejercicio3.txt y detalle lo siguiente:

a) ¿Qué tipos de algoritmos podría utilizar para resolver el problema? ¿Cuáles de ellos le aseguran que calcula el número mínimo de tubos?
b) Escriba un programa que calcule qué tubos hay que cargar en el camión para entregárselos al cliente.

Ejercicio 4: (2,5 puntos).

Para cierta aplicación se quiere disponer de una operación de la clase Grafo que permita conocer si el grado de un vértice es divisible por 3 o no.

Cree el archivo Grafo.java y ponga como comentario al principio del archivo //Ejercicio4.

Conteste a las preguntas aquí plasmadas dentro de ese archivo.

1. a) Deberá escribir en la clase Grafo un método con la cabecera que se indica a continuación. Utilice como apoyo la clase Grafo que se incluye a continuación.
2. b) Escriba al menos 3 pruebas que permitan probar el método del apartado anterior sobre el grafo. Para cada prueba indique los siguientes elementos: Objetivo de la prueba, Valores para la prueba, Preámbulo, Postámbulo, Resultado esperado.

```
public int GradoDivisible3 (int grado) {}
```

Nombre y apellidos:
NP:
Firma:

Código de la implementación de la clase Grafo:

```
class Grafo {
    public Grafo(int NumVertices) {}
    public void añadirArista(Arista arista) throws
GrafoException {}
    public void añadirVertice(Vertice ver) throws
GrafoException{}
    public void eliminarArista(Arista arista) throws
GrafoException {}
    public void eliminarVertice(Vertice ver) throws
GrafoException{}
    private int posicionVertice(Vertice ver) {}
    public Vertice[] listaVertices() {}
    public Arista[] listaAristas() {}
    private int numeroDeAristas() {}
    private int numeroDeVertices() {}
}
public class Vertice {
    private String nombre;
    public Vertice(String nombre) {}
    public String getNombre() {}
    public boolean igual(Vertice v) {}
}
public class Arista
{
    private Vertice v1;
    private Vertice v2;
    private int peso;
    public Arista(Vertice v1, Vertice v2, int peso) {}
    public Vertice getV1() {}
    public Vertice getV2() {}
    public int getPeso() {}
}
```