TST - Programación II - Segundo Parcial                                                11-11-2016

Ejercicio 1

Dado el String ABCDEFGHIJ obtenido por un algoritmo que recorre un Árbol Binario de forma pre-orden. Sabiendo que el nodo A es la raíz del árbol y los nodos C, E, H, J son nodos hojas, escriba un programa que inicialmente cargue en memoria de forma estática la estructura del árbol binario y el método que se utilizó para recorrerlo.

Entrada

Estructura de Árbol Binario y String que resultó del recorrido del Árbol en forma post-orden.

Salida

Imprimir el recorrido pre-orden del Árbol Binario.

Ejemplos

|  |  |
| --- | --- |
| Entrada | Salida |
| La entrada del algoritmo es el árbol binario y el string previamente obtenido:  ABCDEFGHIJ | A  B  C  D  E  F  G  H  I  J |

Aclaraciones

Crear la estructura TAD que permita cargar el Árbol Binario en memoria.

La estructura del árbol binario puede variar dependiendo de cómo se ordenen los nodos, pero debe respetar la posición de los nodos conocidos, como A (raíz) y C, E, H, J que son hojas.

Ejercicio 2

Dado un grid (matriz) que representa un mace (laberinto) diseñar un algoritmo que permita recorrerlo, y llegar hasta la meta. Donde el inicio es la letra I y la salida es la letra S. Transitando por los . (puntos). Los # (numeral) son obstáculos, que no se pueden traspasar. Listar los pasos a seguir para llegar al objetivo (no importa el orden).

Entrada

Una matriz, compuesta por . y #, que indica una I de inicio, y una S de salida.

Salida

Pasos a seguir para salir, sin importar el orden.

Ejemplos

|  |  |
| --- | --- |
| Entrada | Salida |
| I . .  . # .  . . S | derecha, derecha, abajo, abajo. |

Aclaraciones

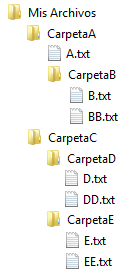
Recuerden que si tenemos una matriz y planteamos un recorrido, dadas las restricciones del ejercicio, esta matriz podría asemejarse a un grafo, y por lo tanto podemos aplicar técnicas de grafos para recorrer la matriz.

Los ejemplos de entrada están separados por espacios por cuestiones de formato, pero la entrada será una línea sin espacios en medio, compuesta por punto y numeral.

Si en algún momento se hace un paso incorrecto, no quita puntaje que se liste.

Ejercicio 3

Dada la estructura de Carpetas y Archivos de la siguiente imagen, escriba un programa para cargar dicha estructura estática en memoria y un algoritmo que permita listar el contenido de cada carpeta.



Entrada

Estructura de Árbol Binario que representa la estructura de carpetas y archivos de la figura anterior.

Salida

Se debe imprimir como salida el nombre de la carpeta o archivo que se encuentra dentro de cada carpeta.

Ejemplos

|  |  |
| --- | --- |
| Entrada | Salida |
| La entrada del algoritmo es el árbol binario. | Mis Archivos  CarpetaA  CarpetaC  CarpetaB  A.txt  CarpetaC  CarpetaD  B.txt  BB.txt  D.txt  DD.txt  E.txt  EE.txt |

Aclaraciones

La estructura de Carpetas y Archivos posee la estructura de un Árbol Binario.

La carpeta “Mis Archivos” representa la raíz del Árbol.

Sabiendo que la estructura del TAD Cola es la siguiente:

|  |
| --- |
| /\*\*  \* Clase Nodo.  \*/  public class Nodo {     public string valor;     public Nodo siguiente;     public Nodo(string valor) {         this.valor = valor;         this.siguiente = null;     }     // toString() retorna el valor del nodo.     @Override     public String toString() {         return this.valor;     }  }  \*\*  \* Clase cola.  \*/  public class Cola {     private Nodo primero;     private Nodo ultimo;     private int tamaño;     public Cola() {         this.primero = null;         this.ultimo = null;         this.tamaño = 0;     }     public boolean vacia() {         return this.primero == null;     }     // enqueue     public void encolar(string valor) {         Nodo nuevo = new Nodo(valor);         if (vacia()) {             this.primero = nuevo;             this.ultimo = nuevo;         } else {             this.ultimo.siguiente = nuevo;             this.ultimo = nuevo;         }         this.tamaño++;     }     // dequeue     public int desencolar() {         if (vacia()) {             return -1;  // cola vacía         } else {             int aux = this.primero.valor;             this.primero = this.primero.siguiente;             this.tamaño--;             return aux;         }     }     public void imprimir() {         if (vacia()) {             System.out.println("Cola vacia");         } else {             Nodo aux = this.primero;             for(int i = 0; i<this.tamaño; i++) {                 System.out.println(aux.valor);                 aux = aux.siguiente;             }         }     }  } |