NATIONAL UNIVERSITY OF DISTANCE EDUCATION

Estrategias de Programación y Estructuras de Datos

Idioma: EN

INSTRUCTIONS:
Programming Strategies and Data Structures. June 2025 · 2nd Week

Exercises that require programming must be done in Java, using the course ADTs (the interfaces for these ADTs are attached to this statement).

Cost-calculation exercises require explicitly stating the problem size. If this is not done, the answer will not be evaluated.

All answers must be justified; answers without justification will not be evaluated.

ADT Interfaces

CollectionIF
```java
public interface CollectionIF {
public int size();
public boolean isEmpty();
public boolean contains(E e);
public void clear();
}
```

SequenceIF
```java
public interface SequenceIF extends CollectionIF {
public IteratorIF iterator();
}
```

ListIF
```java
public interface ListIF extends SequenceIF {
public E get(int pos);
public void set(int pos, E e);
public void insert(int pos, E elem);
public void remove(int pos);
}
```

StackIF
```java
public interface StackIF extends SequenceIF {
public E getTop();
public void push(E elem);
public void pop();
}
```

QueueIF
```java
public interface QueueIF extends SequenceIF {
public E getFirst();
```

```
public void enqueue(E elem);
public void dequeue();
}
```

TreeIF
```java
public interface TreeIF extends CollectionIF {
public E getRoot();
public boolean isLeaf();
public int getNumChildren();
public int getFanOut();
public int getHeight();
public IteratorIF iterator(Object mode);
}
```

GTreeIF
```java
public interface GTreeIF extends TreeIF {
enum IteratorModes { PREORDER, POSTORDER, BREADTH }
public void setRoot(E e);
public ListIF> getChildren();
public GTreeIF getChild(int pos);
public void addChild(int pos, GTreeIF e);
public void removeChild(int pos);
}
```

BTreeIF
```java
public interface BTreeIF extends TreeIF {
enum IteratorModes { PREORDER, POSTORDER, BREADTH, INORDER, RLBREADTH }
public BTreeIF getLeftChild();
public BTreeIF getRightChild();
public void setRoot(E e);
public void setLeftChild(BTreeIF child);
public void setRightChild(BTreeIF child);
public void removeLeftChild();
public void removeRightChild();
}
```

BSTreeIF
```java
public interface BSTreeIF> extends TreeIF {
enum IteratorModes { DIRECTORDER, REVERSEORDER }
enum Order { ASCENDING, DESCENDING }

public BSTree getLeftChild();
public BSTree getRightChild();
public void add(E e);
public void remove(E e);
public Order getOrder();
}
```

Question 1

Practice question.
It is required to program an operation:
```java
ListIF getTasksBetweenDates(int dI, int dF)
```
that returns the list of tasks to be performed between dates dI and dF, both included, specified by the parameters and stored in the future task scheduler.
As a precondition, assume that dI < dF.

a) (1 point) Implement getTaskBetweenDates(dI, dF) so that it is independent of the structure chosen to implement the task scheduler.

b) (1 point) Compute the worst-case asymptotic time cost of getTaskBetweenDates(dI, dF).

STUDENT RESPONSE:

a)
```
public ListIF<TaskIF> getTasksBetweenDates(int dI, int dF) {
ListIF<TaskIF> result = new ArrayList<>();
IteratorIF<TaskIF> it = tasks.iterator();
while (it.hasNext()) {
TaskIF task = it.next();
int date = task.getDate();
if (date >= dI && date <= dF) {
result.insert(result.size(), task);
}
}
return result;
}
```

b)
The problem size is the number of tasks stored in the scheduler, denoted as n.
The worst-case asymptotic time cost is O(n).

Question 2

Analyze the following code fragments and determine their worst-case asymptotic time cost:

a) (1.5 points)

```java
int i = 1;
while (i < n) {
System.out.println(i);
i *= 2;
}
```

b) (1.5 points)

```java
public static int dum(int n) {
if (n == 0) {
return 0;
} else if (n <= 3) {
return 1;
} else {
return dum(n-1) + dum(n-2) + dum(n-3);
}
}
```

STUDENT RESPONSE:

(Sin respuesta)

Question 3

Suppose a turn-based game with several players whose number can decrease during the game when someone is eliminated during their turn. Each player performs an action and then passes the turn to the next player, in circular order.

You are asked to program a Data Type that lets us manage the order in which players take their turns. For this, two operations are needed:
- pasarTurno(): advances the turn to the next player.
- eliminarJugador(): removes the current player from the game.

a) (0.5 points) Indicate which data structure studied in the course would be the most appropriate, so that the cost of both operations is optimized. Also indicate how the current player can be identified in that structure.

b) (1.5 points) Implement pasarTurno() and eliminarJugador().

STUDENT RESPONSE:

a) A circular linked list is the most appropriate data structure. The current player is identified by a reference to the current node.

b)

```
public void pasarTurno() {
current = (current + 1) % players.size();
}
public void eliminarJugador() {
players.remove(current);
if (current >= players.size()) current = 0;
}
```

Question 4

Given the following tree, indicate the sequence of nodes visited by preorder, postorder, and breadth-first traversals (from left to right).

1
/ | \n 2 3 4
/ \ |
5 6 7

Explain how each traversal is performed.

STUDENT RESPONSE:

Question 4:

a) Inserting 3, 1, 2: Unbalanced tree has root 3 with left child 1 (right child 2). Requires left-right rotation. First rotate left at 1 (making 2 parent of 1), then rotate right at 3. Resulting AVL tree: root 2 with left child 1 and right child 3.

b) Inserting 3, 2, 1: Unbalanced tree has root 3 with left child 2 (left child 1). Requires single right rotation at 3. Resulting AVL tree: root 2 with left child 1 and right child 3.

c) Inserting 7, 15, 10: Unbalanced tree has root 7 with right child 15 (left child 10). Requires right-left rotation. First rotate right at 15 (making 10 parent of 15), then rotate left at 7. Resulting AVL tree: root 10 with left child 7 and right child 15.

d) Inserting 11, 22, 35: Unbalanced tree has root 11 with right child 22 (right child 35). Requires single left rotation at 11. Resulting AVL tree: root 22 with left child 11 and right child 35.