מחלקת חוליה על פי משרד החינוך –

```
class Node<T>
 {
    private T value;
    private Node<T> next;
    public Node(T value)
    {
      this.value = value;
      this.next = null;
    }
    public Node(T value, Node<T> next)
    {
      this.value = value;
      this.next = next;
    }
    public T GetValue()
    {
      return this.value;
    }
    public Node<T> GetNext()
    {
      return this.next;
    }
    public bool HasNext()
    {
      return this.GetNext() != null;
    }
    public void SetValue(T value)
    {
      this.value = value;
}
```

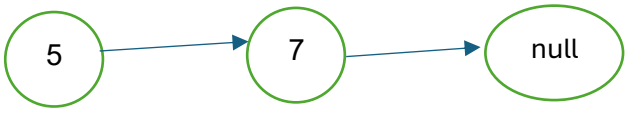
```
public void SetNext(Node<T> next)
{
   this.next = next;
}

public override string ToString()
{
   Node<T> pos=this;
   string str="[";
   while (pos.HasNext())
   {
      str += pos.value + ",";
      pos = pos.next;
   }
   str += pos.value + "]";
   return str;
 }
}
```

פעולות פנימיות שקיימות על חוליות :

תיאור הפעולה	פעולה בונה א' public Node(T value)	פעולה בונה ב' Public Node(T value , Node<T> next)	GetValue	GetNext	HasNext	SetValue	SetNext	toString
	הפעולה מקבלת טיפוס מסוג החוליה ומבצעת השמה לערך של החוליה	הפעולה מקבלת טיפוס של הערך של החוליה הנוכחית ומי החוליה הבאה	הפעולה מחזירה את הערך של החוליה הנוכחית	הפעולה מחזירה את החוליה הבאה	הפעולה מחזירה אמת אם קיימת חוליה שהיא לא ריקה בחוליה הבאה	הפעולה מקבלת טיפוס מסוג החוליה ומבצעת לו השמה לערך של החוליה	הפעולה מקבלת טיפוס מסוג חוליה ומבצעת לו השמה להבא של החוליה	הפעולה מחזירה את החוליה הנוכחית כמחרוזת המהווה ייצוג מחרוזתי של הערך של החוליה

כדי להבין מבנה נתונים זה עלינו להבין מה התכונות שיש לטיפוס זה -
לכל חוליה יש ערך – מכל סוג שנגדיר
לכל חוליה יש מצביע לחוליה הבאה – במידה ואין חוליה הבאה יוגדר null

זאת אומרת שאם יש לי חוליה שיש לה ערך של המספר 5 והיא מצביעה לחוליה נוספת שלה יש ערך 7 והיא מצביעה על null אפשר לומר ש2 החוליות שלא ריקות יוצרות לי סוג של שרשרת (חד כיוונית) אמנם קצרה אך שרשרת:



במילים אחרות חוליות שיש בניהם קשרים יוצרות לי שרשראות
כל שרשרת מתחילה בחוליה כלשהית עם ערך ומסתיימת ב null

עקרונות:

מעבר על שרשרת חוליות באופן פשוט

```
0 references
public static void printNode( Node<int> node)
{
    while(node != null) // רצים על השרשרת כל עוד היא לא ריקה
    {
        Console.WriteLine(node.GetValue()); // מדפיסים כל ערך בכל חוליה בשרשרת
        node = node.GetNext(); // קידום לשרשרת לעבור לחוליה הבאה
    }
}
```

"הבעיה" בקוד היא שבסוף ההרצה של התכנית אנחנו נקבל ב node את null
מהסיבה שהמצביע לחוליה שלו עבר לסוף , ל null

במידה ונתקל בשאלות כמו "עליכם לשמור על מבנה השרשרת המקורי"
עלינו להשתמש ב"עקרון המצביע"
כאשר אנחנו יוצרים מצביע חדש ורק אותו מקדמים בעוד את המצביע המקורי אנחנו לא מקדמים

```
0 references
public static void PrintNode2(Node<int> node)
{
    Node<int> temp = node; // יצירת מצביע חדש בזיכרון
    while(temp != null)
    { // מעבר וקידום בטוח על משתנה אחר בולל קידום
        Console.WriteLine(temp.GetValue());
        temp = temp.GetNext();
    }
}
```

פעולות נוספות שיכולות להיות לעזר-

פעולה העתיקה שרשרת חוליות:

```
0 references
private static Node<int> CopyNodes(Node<int> head)
{
    if (head == null) return null;

    Node<int> newHead = new Node<int>(head.GetValue());
    Node<int> currentOriginal = head.GetNext();
    Node<int> currentCopy = newHead;

    while (currentOriginal != null)
    {
        Node<int> newNode = new Node<int>(currentOriginal.GetValue());
        currentCopy.SetNext(newNode);

        currentOriginal = currentOriginal.GetNext();
        currentCopy = currentCopy.GetNext();
    }

    return newHead;
}
```

פעולה ההופכת שרשרת חוליות:

```
0 references
public static Node<int> Reverse(Node<int> head)
{
    Node<int> prev = null; // החוליה הקודמת
    Node<int> current = head; // החוליה הנוכחית
    Node<int> next = null; // החוליה הבאה

    while (current != null)
    {
        next = current.GetNext(); // שמירה על החוליה הבאה
        current.SetNext(prev); // הפיכת הכיוון
        prev = current; // התקדמות
        current = next; // התקדמות
    }

    return prev; // החוליה החדשה של ראש הרשימה
}
```