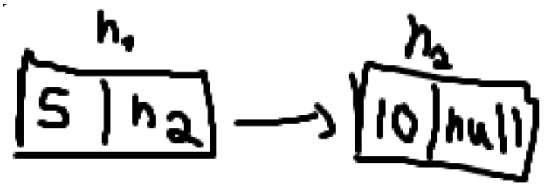
Node

Node <Integer> name1 = new node<Integer>(5,null) פעולה בונה

Node <Integer> name2 = new node<Integer>(10,null)

n1.setNext(n2);



n1.setValue(30) – sets value of n1 to 30

n1.setNext(new Node<Integer>(40,null)) – *sets the node after n1 to(40,null)*

n1.getValue //takes the 1st place in n1

n1.getNext().setValue(77) - sets the value of the thing after n1 to 77

scans everything including the last one

```
while(n1!=null){
        n1=n1.getNext();
}
```

Scans everything not including the last one

```
While(hasNext()){
        n1 = n1.getNext();
}
```

כתוב פעולה המקבלת רשימה המחזירה רשימה שהיא העתק של המקורית בלי העתקים

```
Public static Node<Integer> nod(Node <Integer> ls) //ls is the name
    Node <Integer> pos =ls; //pos is a copy of ls
    Node<Integer> ls1 = new Node <Integer> (pos.getValue())
    While(pos!=null){
        If(!isInList(ls1, pos.getValue()){ /isInList is a func
            Last.setNext(new Node<Integer>(pos.getvalue());
            Last = last.getNext();
        }
    pos=pos.getNext();
```

```
                {
                        prev=l;
                        l=l.getNext();

                }
        }
        return first;
}

//      o(n*n) הפעולה מקבלת רשימה ומספר ומוציאה מהרשימה את כל האיברים ששוים למספר
// תרגיל 5 בסדר גודל ריבועי //
public static Node<Integer> removeNumItems2(Node<Integer> l, int num)
{
        Node<Integer> first=l;
        while(l!=null)
        {
                if(l.getValue()==num)
                        if (first==l)
                        {
                                first=first.getNext();
                                l.setNext(null);
                                l=first;
                        }
                        else
                        {
                                // סיבוכיות ריבועית זה לינארי בלולאה לינארית
                                Node<Integer>p=first;
                        while(p.getNext()!=l)// מציאת החולייה הקודמת
                                p=p.getNext();
                        p.setNext(l.getNext());
                        l.setNext(null);
                        l=p.getNext();

                                }
                        else
                        {

                                l=l.getNext();

                        }
        }
        return first;
}


//      o(n*n) הפעולה מקבלת רשימה ומחזירה רשימה חדשה שבה הערכים ברשימה הם ללא כפילויות
// תרגיל 6 בסדר גודל ריבועי //
public static Node<Integer> noDoubleItemNewList(Node<Integer> l)
{

        Node<Integer> newList = new Node<Integer>(l.getValue());    // איבר ראשון ברשימה
החדשה

        Node<Integer> p=newList;
        while(l!=null)
        {
                if (!isIn(newList, l.getValue()))
```

```
                {
                        p.setNext(new Node<Integer>(l.getValue()));
                        p=p.getNext();
                }
                l=l.getNext();
        }
        return newList;
}

//      o(n) הפעולה מקבלת רשימה ומספר ומחזירה האם המספר נמצא ברשימה
public static boolean isIn(Node<Integer> l, int num)
{

        while(l!=null)
        {
                if(l.getValue()==num)
                        return true;
                l=l.getNext();
        }
        return false;
}


//      o(n*n) הפעולה מקבלת רשימה ומעדכנת את הרשימה לרשימה ללא כפילויות
// תרגיל 6 בסדר גודל ריבועי //
public static void noDoubleItem(Node<Integer> l)
{
        Node<Integer> prev=l, p=l;
        while(l.hasNext())
        {
                p=l.getNext();
                prev=l;
                while(p!=null)
                {
                        if (p.getValue()==l.getValue())
                        {
                                prev.setNext(p.getNext());
                                p.setNext(null);
                                p=prev.getNext();
                        }
                        else
                        {
                                prev=p;
                                p=p.getNext();
                        }
                }
                l=l.getNext();
        }
}


//      o(1) הפעולה המגרילה 50 מספרים דו ספרתיים ומחזירה רשימה ללא חזרות של המספרים
המוגרלים שספרותיהן שונות זו מזו
// תרגיל 7 בסדר גודל קבוע - לולאה עד 50
```

תורים-

להוסיף null לסוף התור –

```
q.insert(null);
```

יצירת תור ריק –

```
Queue<Integer>q2 = new Queue<Integer>();
```

הוספת מספר לתור ריק –

```
q.insert(x);
```

הוצאה את האיבר הראשון של תור –

```
Integer x = q.remove();
```

לולאה שעוברת על כל התור שקיבלנו –

```
while(q.head()!=null){
```

כדי להתקדם בתוך תור –

```
Integer x = q.remove() ;
q.insert(x);
```

שאתה יוצא מהלולאה והגעת ל null כדי להוציא אותו –

```
q.remove();
```