**תבניות ברשימה**

**סריקת רשימה מתחילתה עד סופה**

**1**. **סכום איבריה.** O(N)

public static int sum(Node<Integer> lst)

{

int sum = 0;

while(lst != null)

{

sum = sum + lst.getValue();

lst = lst.getNext();

}

return sum;

}

**2. נמצא ברשימה**  O(N)

public static boolean found(Node<Integer> lst, int n)

{

while lst!= null)

{ if(lst.getValue()==n)

return true;

lst = lst.getNext();

}

return false;

}

**3. מיקום איבר ברשימה :** O(N)

public static Node< Integer > where (Node< Integer > lst, int n)

{

while )lst!= null)

{ if(lst.getValue()==n)

return lst;

lst = lst.getNext();

}

return lst;

}

**בדיקת איברים סמוכים**

**4 .דוגמה : האם רשימה ממוינת**  O(N)

public static boolean isSorted(Node<Integer> lst)

{

if (lst!=null)

{

int prev=lst.getValue();

lst= lst.getNext();

while( lst!= null)

{

int now=lst.getValue(); if(now<=prev)

return false;

prev=now;

lst = lst.getNext();

}

}

return true;

}

**עבור כל איבר בדיקת העוקבים לו**

**5. דוגמה: ניפוי רשימה מכפילויות .** O(N^2)

public static void removeDuplications(Node<Integer> lst)

{

int n;

Node< Integer> pos1 = lst;

Node< Integer> pos2;

while(pos1 != null)

{

n = pos1.getValue();

Node< Integer> prev=pos1;

pos2 = pos1.getNext();

while(pos2 != null)

{

if(pos2.getValue() == n)

{

prev.setNext(pos2.getNext());

pos2.setNext(null);

pos2=prev.getNext();

}

else

pos2 = pos2.getNext();

}

pos1 = pos1.getNext();

}

}

**סריקת 2 רשימות- עבור כל איבר ברשימה אחת, סריקת כל איברי הרשימה השנייה.**

**6א. דוגמה: בניית רשימת חיתוך (איברים המשותפים לשתי רשימות).** O(N\*M)

**הנחה: בכל רשימה האיברים שונים. הרשימה נבנית בסדר הפוך להכנסה.**

public static Node<Integer> chituch(Node<Integer> l1, Node<Integer> l2)

{

Node<Integer> list=null;

while(l1 != null)

{

int x =l1.getValue();

Node<Integer> pos2 = l2;

while(pos2 != null)

{

if (pos2.getValue() == x)

list=new Node<Integer>(x,list);

pos2 = pos2.getNext();

}

l1=l1.getNext();

}

return list ;

}

**6ב**.  **דוגמה: בניית רשימת חיתוך (איברים המשותפים לשתי רשימות). –שימוש בנמצא ברשימה**

**הנחה: בכל רשימה האיברים שונים.**

public static Node<Integer> chituch(Node<Integer> l1, Node<Integer> l2)

{

Node<Integer> list=null;

while(l1 != null)

{

int x =l1.getValue();

if (found (l2,x))

list=new Node<Integer>(x,list);

l1=l1.getNext();

}

return list ;

}

**פעולות שימושיות:**

**7. מקום האיבר הקודם לאיבר שההפנייה אליו נתונה ב pos ברשימה:**

public static Node<Integer> **prev** (Node<Integer> lst, Node<Integer> pos)

{

Node<Integer> prv=lst;

if (prv==pos)

return null; // קודם לאיבר הראשון

while ( prv.getNext()!=pos)

prv=prv.getNext();

return prv;

}

**8.מקום האיבר האחרון ברשימה:**

public static Node<Integer> **last** (Node<Integer> lst)

{

Node<Integer> prv=lst;

if (prv==null)

return prv;

while (prv.getNext()!=null)

prv=prv.getNext();

return prv;

}

**בניית רשימה**

**9. פעולה היוצרת רשימה של מספרים אקראיים הרשימה נוצרת בסדר הפוך**

**public** **static** Node<Integer> buildRandom(**int** n, **int** low, **int** up)

{

**int** num;

Node<Integer> list = **null**;

**for** (**int** k=1; k<=n; k++)

{

num = (**int**)(Math.*random*()\*(up-low+1))+low;

list = **new** Node<Integer>(num,list);

}

**return** list;

}

**10. פעולה היוצרת רשימה של מספרים אקראיים הרשימה נוצרת בדיוק לפי סדר ההכנסה**

**public** **static** Node<Integer> build2(**int** n, **int** low, **int** up)

{

Node<Integer> list =**null**;

Node<Integer> p=list;

**for** (**int** k=1; k<=n; k++)

{

int num = (**int**)(Math.*random*()\*(up-low+1))+low;

Node<Integer> p2=**new** Node<Integer>(num,**null**);

**if** ( list==**null**)

{

list=p2;

p=list;

}

**else**

{

p.setNext(p2);

p=p.getNext();

}

}

**return** list;

{