**תרגול 1**

[**Floyd's cycle-finding algorithm**](http://en.wikipedia.org/wiki/Cycle_detection#Tortoise_and_hare), also known as ***tortoise*** *and* ***hare*** *algorithm*.   
It is one of the **simple cycle detection algorithm**.  
The idea is to have **two pointers** to the list (Singly Linked List) and move them at **different speeds**.

Move the **first/slow** **pointer (also called tortoise)** forward by **1** node/step and the **second**/**fast pointer (also called hare)** by **2** nodes.

Floyd Cycle detection algorithm consists of three parts:

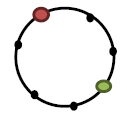
* Cycle detection in linked list
* Finding start of the cycle/loop.
* Measuring length of the cycle/loop.

**Problem 1: To check for the existence of the cycle.**

**בעיה 1.1 :** **בעיית חיפוש מעגל ברשימה מקושרת**

* **if** the linked list has a loop they will ***definitely*** meet.

**Proof**:



נוכיח כי ארנב וצב נפגשים כאשר רשימה מכילה מעגל:

נניח כי n – מספר איברים ברשימה, k – מרחק מנקודת ההתחלה (עיגול אדום) עד לנקודת המפגש של הרובוטים (עיגול ירוק),

i – מספר צעדים שהצב עושה, 2\*i – מספר הצעדים שהארנב עושה. לפי כך, ניתן לכתוב כי:

i = np + k, 2i = nq + k

כאשר p – מספר סיבובים שהצב עושה ו-q – מספר סיבובים שהארנב עושה. מכאן מקבלים:

2np + 2k = nq + k

או

k = n(q – 2p) (\*)

מכאן נובע ש-k הוא כפולה של n, כלומר הצב והארנב נפגשים והמפגש יתקיים בנקודת ההתחלה.

מספר סיבובים מינימלי שמקיים את המשוואה (\*): כאשר q=2, p=1 אז מקבלים k=0.

* **else** either of the two pointers (or their next) will become **null**.

**בעיה1.2 : בעיית חיפוש מעגל ברשימה מקושרת עם זרוע**

****

נוכיח כי גם במקרה כזה ארנב וצב נפגשים כאשר רשימה מכילה מעגל:

נניח כי n – אורך מעגל (מספר איברים במעגל), k – מרחק מנקודת ההתחלה עד לנקודת המפגש של הרובוטים,

m – אורך הזרוע (מספר איברים בזרוע), i – מספר צעדים שהצב עושה, 2\*i – מספר הצעדים שהארנב עושה. לפי כך, ניתן לכתוב כי:

i = m + np + k, 2i = m + nq + k

כאשר p – מספר סיבובים שהצב עושה ו-q – מספר סיבובים שהארנב עושה. מכאן מקבלים:

2m + 2np + 2k = m + nq + k

m + k = n(q-2p) (\*\*)

במשוואה (\*\*) m,n הם מאפיינים של הרשימה הנתונה. ניתן להראות שקיים לפחות סט אחד של k,p,q שמקיים את המשוואה:

אם **p=0 ו-q=m** אז m + k = n\*m  **k = n\*m - m** ← ←

= n\*m i = m + n\*0 + n\*m – m כמובן שה-i הזה לא קטן ביותר. כלומר, צב וארנב יכולים להיפגש לפני כבר כמה פעמים. עם זאת, מכיוון שאנו מראים שהם נפגשים בשלב מסוים לפחות פעם אחת אנו יכולים לומר שההשערה היא נכונה. אז הרובוטים יפגשו אם נעביר אחד מהם צעד אחד והשני 2 צעדים בכל פעם.

מהמשוואה (\*\*)k = n(q-2p)- m :, אז נקודת המפגש נמצא במרחק n-m כאשר m<n. אם m>n, m=m%n

**Pseudocode // Problem 1: To check for the existence of the cycle.**

**Time Complexity: O(n)**

**hasLoop**(Node first)

**if** (first = **null**) **return** **false** // list does not exist, so no loop either.

end **if**

Node slow, fast // create two references.

slow← first // make both references to the start of the list.

fast ← first

**loop** (**true**)

slow ←next(slow) // makes 1 step

**if** (next(fast)≠ **null**)

fast←next(next(fast)) // makes 2 steps

**else return** **false** // next node null => no loop.

end **if - else**

**if** (slow=**null** || fast=**null**) **return** **false** // if null, no loop.

end **if**

**if** (slow = fast) **return** **true** // we must have a loop.

end **if**

end **loop**

end **hasLoop**

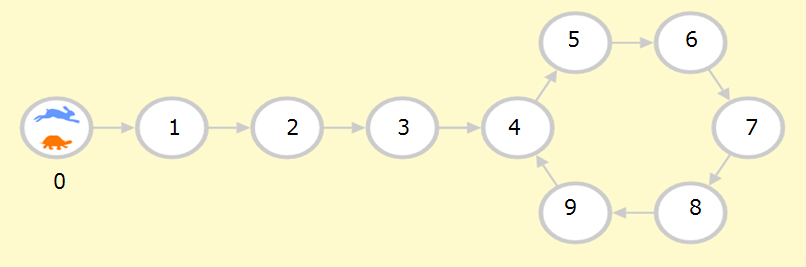
**Problem2 : To find the meeting point.**

**בעיה 2: מציאת נקודת המפגש.**

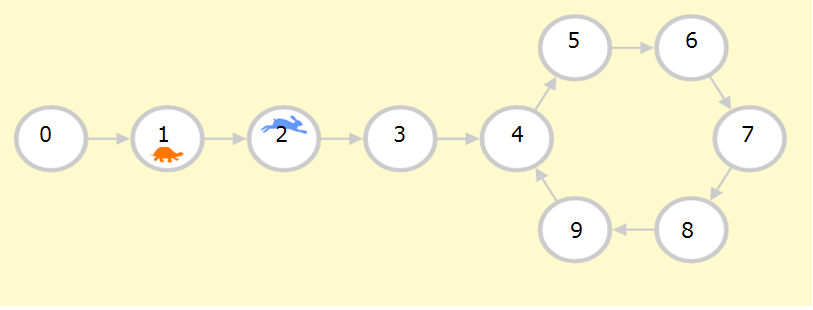
שני רובוטים שמים בתחילת הרשימה, כל אחד זז עם מהירות משלו עד שייפגשו.

**First pointer (*tortoise*) - speed 1 first.next**

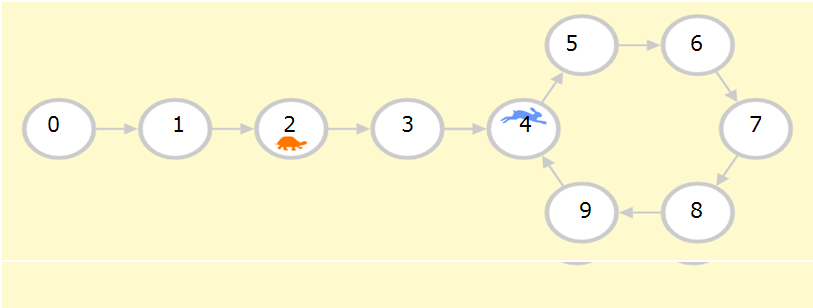
**Second pointer (*hare*) - speed 2 first.next.next**



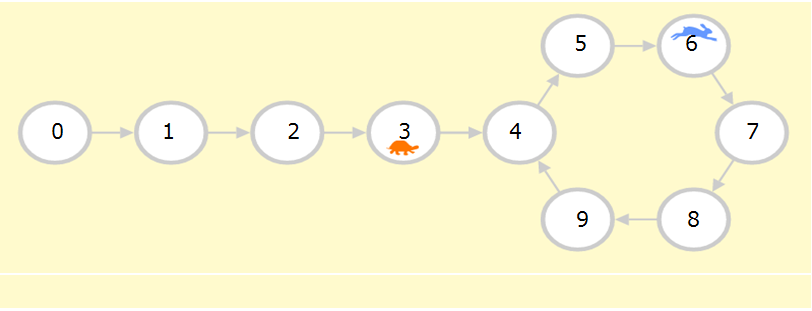
**1**



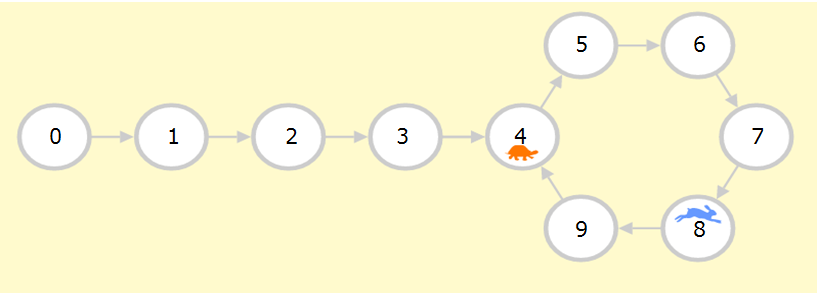
**2**



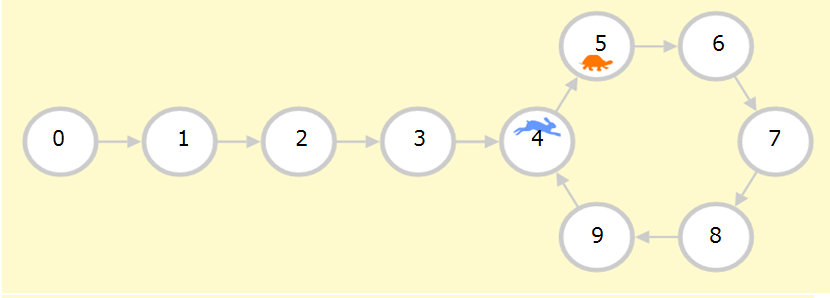
**3**



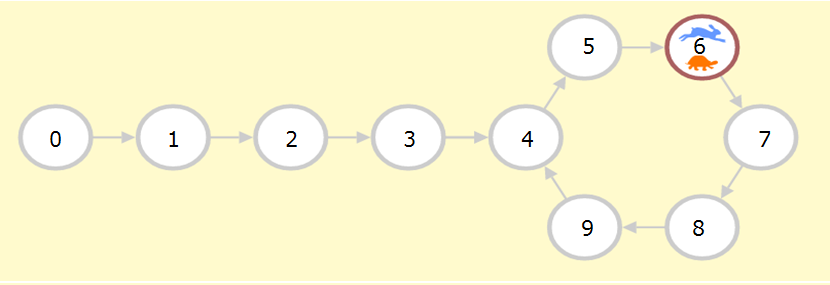
**4**



**5**



**6**



**7**

**Cycle exists → meeting point: 6**

**Pseudocode // Problem 2: To find the meeting point.**

**Time Complexity: O(n)**

**meetNodeInLoop** (Node first)

Node fast, slow

slow← first // make both references to the start of the list.

fast ← first

fast←next(next(fast)) // makes 2 steps

slow ←next(slow) // makes 1 step

**loop** (fast ≠ slow)

fast←next(next(fast)) // makes 2 steps

slow ←next(slow) // makes 1 step

**if** (fast = slow)

**break** fromloop

end **if**

end **loop**

**return** fast

end **meetNodeInLoop**

**Problem 3: To find the loop starting point.**

**בעיה 3: מציאת נקודת התחלה של המעגל**

לאחר ששני הרובוטים נפגשו, נחזיר את הצב בתחילת הרשימה ונשאיר את הארנב בנקודת המפגש (מרחק של k צעדים מתחילת המעגל).

השערה: אם נתן לנוע לשני רובוטים באותה מהירות (צעד אחד לשניהם), הפעם הראשונה שהם יפגשו תהיה נקודת ההתחלה של המעגל.

**First pointer (*tortoise*) - speed 1 first.next**

**Second pointer (*hare*) - speed 1 first.next**

נוכיח את ההשערה:

**meet**

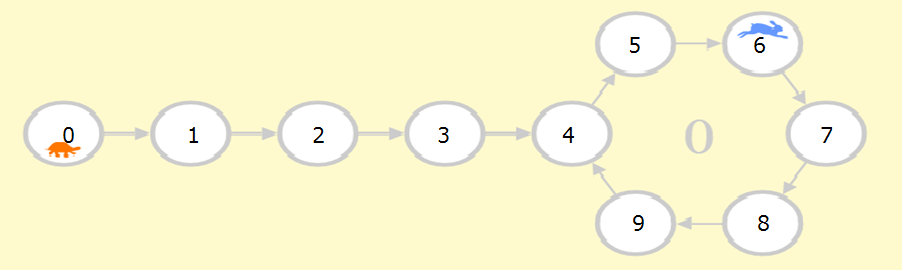
נניח כי ידוע ערך של m ונותנים לרובוטים לנוע m+k צעדים, הצב יגיע לנקודת המפגש, לפי המשוואה (\*\*)m + k = n(q-2p)

מכיוון ש- m + k הוא מכפיל של אורך המעגל n, ארנב יכול לעבור את המעגל (q-2p) פעמים ולחזור לאותה נקודה (k מרחק צעדים מתחילת המעגל).

כעת, במקום לתת להם לנוע m + k צעדים, ניתן להם לנוע רק m צעדים, צב יגיע לתחילת המעגל. והארנב יצטרך לבצע k צעדים להשלמת (q-2p) סיבובים. מכיוון שהארנב התחיל לנוע החל מ- k צעדים מתחילת המעגל, הארנב יגיע לתחילת המעגל.

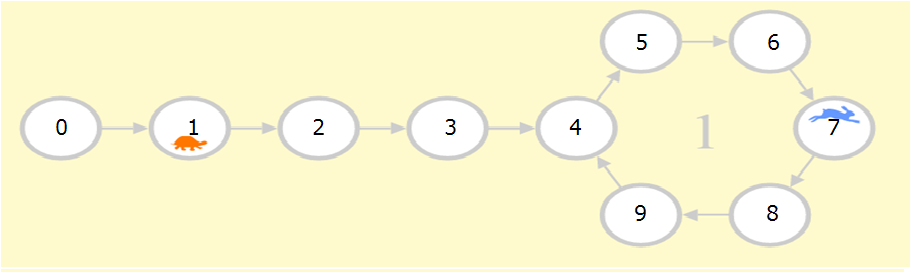
מש"ל

כעת אנו יודעים שמספר הצעדים שיש לבצע לרובוטים עד שהם נפגשים מהוה מרחק מתחילת הרשימה עד לתחילת המעגל, אורך הזרוע m. כמובן שהאלגוריתם אינו יודע מה ערך של m. לכן שני רובוטים ינועו צעד אחד בכל פעם עד שהם ייפגשו. נקודת המפגש תהיה התחלת המעגל ומספר הצעדים יהיה המרחק (m) להתחלת המעגל.

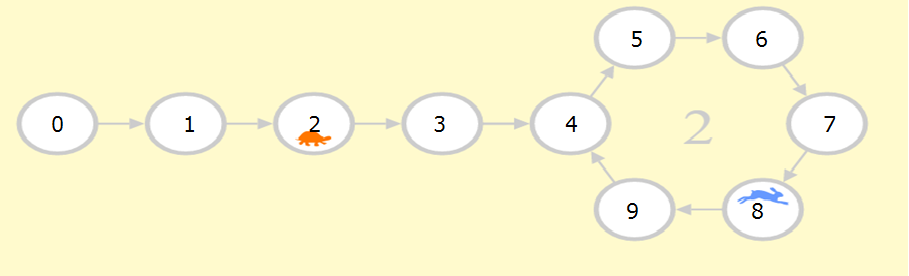


**start**

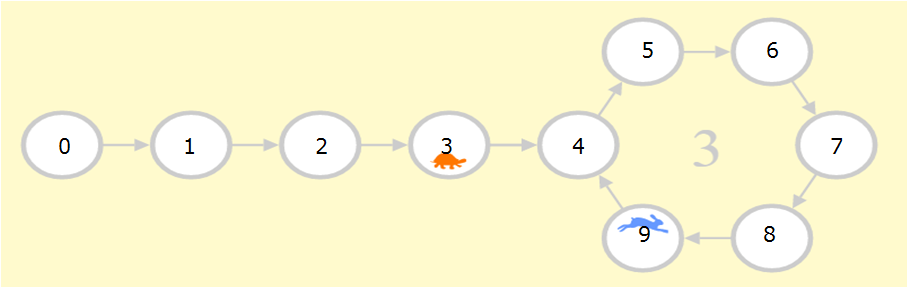
**1**



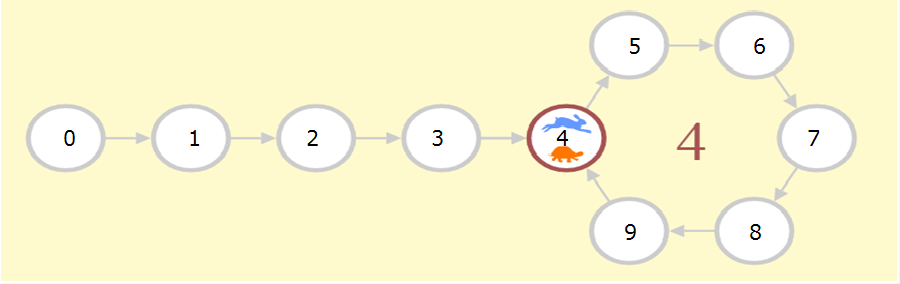
**2**



**3**



**4**



**5**

**Pseudocode // Problem 3: To find the loop starting point.**

**Time Complexity: O(n)**

**startOfLoop**(Node meet)

Node fast, slow

fast←meet // Node fast moves to meet

slow←head // Node slow moves to head

**loop** ( slow ≠ fast ){

fast ← next(fast) // makes 1 step

**if** (slow = next(fast))

**break** from the loop

end **if**

slow ← next(slow) // makes 1 step

end **loop**

**return** slow

end **startOfLoop**

**Problem 4: To find the length of the cycle.**

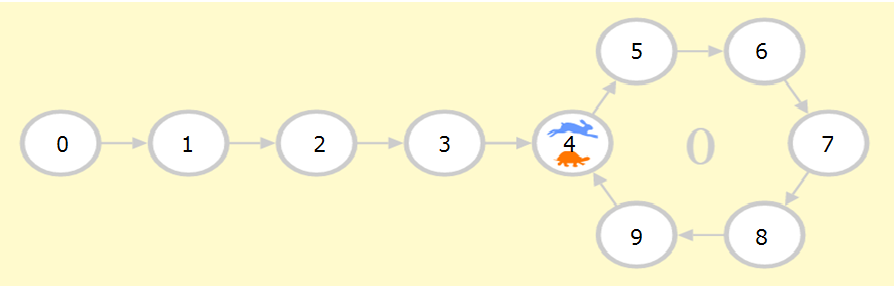
**בעיה 4: מציאת אורך המעגל**

בהנחה שאנו יודעים את אורך הרשימה, נוכל גם לחשב את אורך המעגל ע"י חיסור ה- m (אורך הזרוע) מאורך הרשימה.

לאחר שהרובוטים נפגשו בתחילת המעגל, משארים את הצב בנקודת ההתחלה של המעגל והארנב הולך צעד אחד בכל פעם עד שהוא יפגוש את הצב – מספר צעדים שהארנב יעשה הוא אורך המעגל.

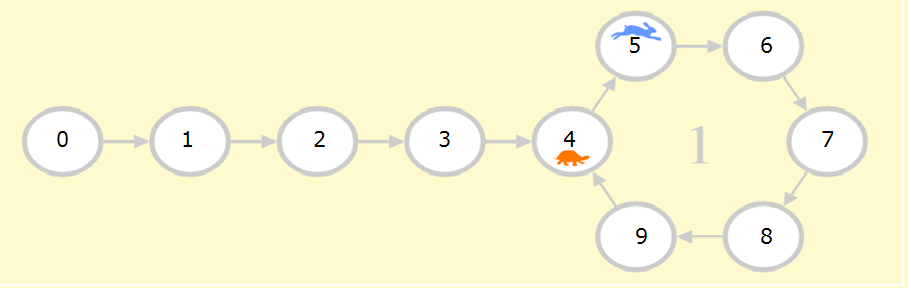
**First pointer (*tortoise*) - speed = 0 (not move)**

**Second pointer (*hare*) - speed = 1 first.next**

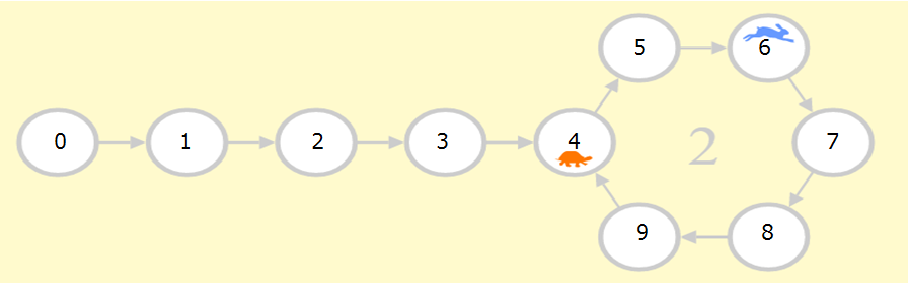


**loopStart**

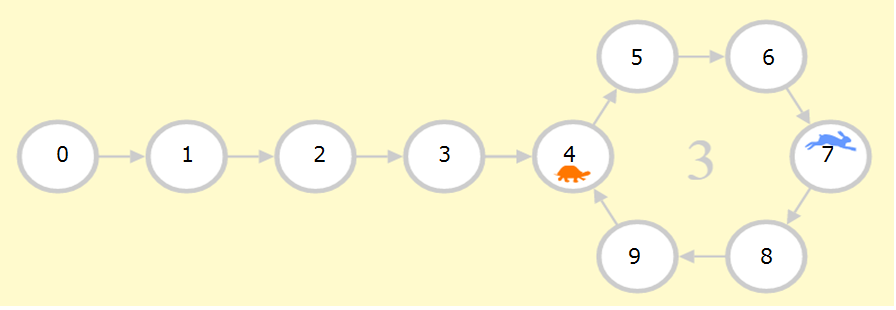
**1**



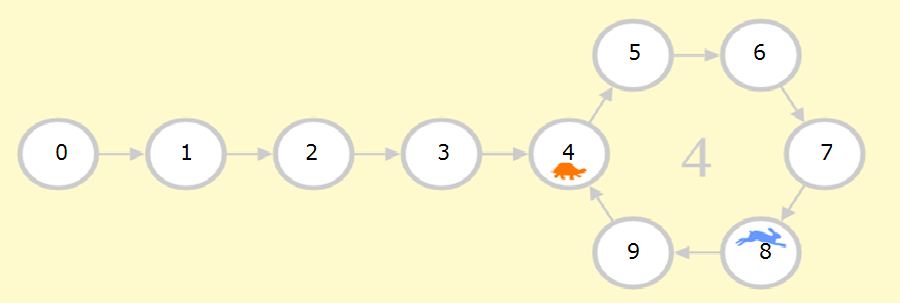
**2**



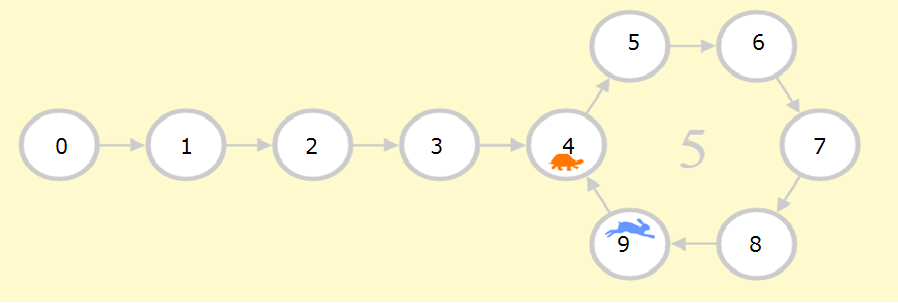
**3**



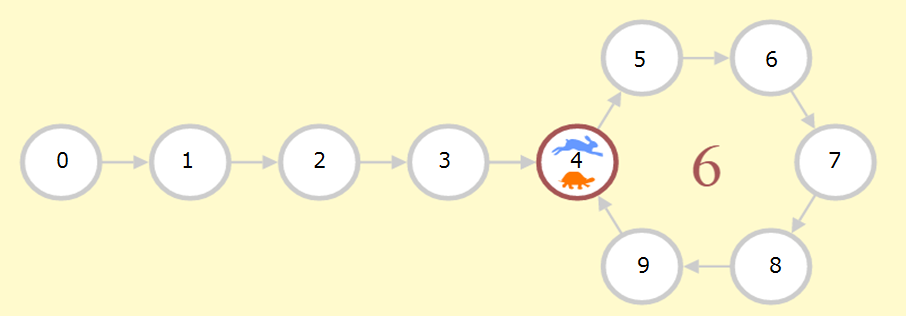
**4**



**5**



**6**



**7**

**Pseudocode // Problem 4 : To find the length of the cycle.**

**Time Complexity: O(n)**

lengthOfLoop(Node loopStart)

**int** length = 0

fast←loopStart // Node fast moves to loopStart

slow← loopStart // Node slow moves to loopStart

fast ← next(fast) // makes 1 step

length= length + 1

**loop** ( slow ≠ fast )

fast ← next(fast) // makes 1 step

length = length + 1

end **loop**

**return** length

end lengthOfLoop