(1)

الف)

```
!Define StackA
!Define StackB

counter = 0
i = 0
while ( i < n):
    height = Input()
    if ( i == 0):
        counter += 1
        StackA.push (height)
    else:
        CanSee = True

        while (!StackA.isEmpty())
            temp = StackA.pop()
            if( height < temp):
                CanSee = False
            StackB.push(temp)

        while (!StackB.isEmpty())
            StackA.push(StackB.pop())

        StackA.push (height)
    i += 1

print (counter)  <-- خروجی
```

Define StackA
Define StackB

```
Function Enqueue(person):
    StackA.push(person)

Function Dequeue:
    if(StackA.isEmpty()):
        print("Queue is empty :)")
    else:
        while(!StackA.isEmpty()):
            StackB.push(StackA.pop())
        result = StackB.pop()
        while(!StackB.isEmpty()):
            StackA.push(StackB.pop())
        return result
```

(۳) برای یک Linked List که head دارد، داریم:

```
p = head

while ( p.next != NULL):
    if( First == p.next):
        return True
    else:
        p = p.next
```

اگر در یک Linked List یک لوپ ایجاد شده باشد، الگوریتم ما True برمی‌گرداند.

---

(۴) باید ۳ تا پوینتر به نام‌های c برای current، p برای Previous و n برای next انتخاب کنیم و داریم:

```
c = head
p = NULL
n = NULL
while ( c != NULL):
    n = c.next
    c.next = p
    p = c
    c = n
head = p
```