فرهاد امان ۹۹۳۱۰۰۲

۱- یک پشته را به عنوان پشته اصلی و پشته دوم را به عنوان پشته کمکی در نظر می‌گیریم.

Enqueue : تنها کافیست دیتای خود را به راس پشته اصلی Push کنیم. $O(1)$

Dequeue : در مرحله اول تا خالی شدن پشته اصلی تمام المان های آن را Pop می‌کنیم و به همان ترتیب در پشته کمکی Push می‌کنیم. بعد از این کار تمام المان های پشته اصلی به صورت reverse در پشته کمکی قرار می‌گیرند. $O(n)$

در مرحله دوم اولین المان پشته کمکی را Pop می‌کنیم به این شکل المان مورد نظر ما به دست خواهیم آورد. $O(1)$

در مرحله سوم المان های باقیمانده در پشته کمکی را Pop کرده و به همان ترتیب در پشته اصلی Push می‌کنیم. تا به حالت اول در آید. $O(n)$

$O(n) + O(1) + O(n) + O(n)$

```
stack main
stack assist
enqueue(x)
    main.push(x)

dequeue
    while main.size > 0
        assist.push(main.pop())
    ans ← assist.pop()
    while assist.size > 0
        main.push(assist.pop())
    return ans
```

فرهاد امان ۹۹۳۱۰۰۶

۲ - دو فرض وجود دارد اول اینکه مانند مثال مطرح شده در سؤال هنگام عبور از یک گره با یک عدد متوجه شماره گره مورد نظر می‌شویم.

راه حل با فرض اول: آرایه‌ای به نام mark تشکیل داده که تمام المان های آن ابتدا ۰ هستند هنگام پیمایش لیست هرگاه از یک گره عبور می‌کنیم مقدار المان با index مساوی با شماره گره را یک می‌کنیم به عنوان مثال هنگام عبور از گره با شماره ۳ المان مربوط به آن array[۳] ← ۱ را انجام می‌دهیم. اگر هنگام عبور از یک گره ۱ در آرایه از قبل ۱ بود لیست دارای حلقه است. اگر پیمایش لیست تا رسیدن به مقدار Null بدون مشکل انجام شد لیست حلقه ندارد.

فرض دوم اینکه گره ها عدد مشخص کننده ندارند. راه حل مانند فرض اول است با این تفاوت که به جای آرایه از یک map استفاده می‌کنیم که key های آن آدرس حافظه گره ها و value های آن یک boolean که نشان دهند عبور از آن گره است. به این علت که آدرس حافظه گره های توانند مقدار بزرگ شود و حرف جزیی در حافظه از آرایه استفاده نمی‌کنیم.

```
array mark[maxsize] = {0}
Pointer curr ← head
while curr ≠ null
    if mark[number of curr] = 1
        return daraye halghe
    else
        mark[number of curr] ← 1
        curr <- curr.next
end while
return bedone halghe
```