اشکان شکیبا (9931030)

سوال اول

پیاده سازی در جاوا:

```
class Queue {
    Stack stack1;
    Stack stack2;

    Queue(){
        stack1=new Stack();
        stack2=new Stack();
    }

    void enqueue(Object object){
        stack1.push(object);
    }

    Object dequeue(){
        if(stack1.isEmpty() && stack2.isEmpty()){
            return null; // queue is empty
        }
        else if(!stack1.isEmpty() && stack2.isEmpty()){
            while(!stack1.isEmpty()){
                stack2.push(stack1.pop());
            }
        }
        return stack2.pop();
    }
}
```

تحلیل زمانی:

پیچیدگی زمانی تابع dequeue، در خط های 5 و 6 آن اتفاق می افتد که در صورت اجرای حلقه در دفعات زیاد، می تواند بار زمانی سنگینی به برنامه تحمیل کند.

برای تحلیل دقیق تر، بد ترین حالت زمانی را در نظر میگیریم، که stack2 هیچ محتوایی نداشته باشد؛ که در این حالت برای انتقال هر یک از اعضای stack1 به stack2، یکبار حلقه اجرا میشود.

اگر تعداد اعضای stack1 را برابر n در نظر بگیریم، خط پنجم n+1 بار و خط ششم n بار اجرا میشود و زمان مورد نیاز برای اجرای دیگر خطوط برنامه، ثابت هایی از مرتبه زمانی یک است.

بنابراین میتوان نوشت که زمان اجرای الگوریتم از مرتبه n ($\Theta(n)$) است.

سوال دوم

پیاده سازی الگوریتم در جاوا:

```
boolean isCycle(Node head){
    Node node1,node2;
    node1=node2=head;

    while(node1!=null && node1.next!=null){
        node1=node1.next;
        node2=node2.next;
        node1=node1.next;
        if(node1.equals(node2)){
            return true;
        }
    }

    return false;
}
```