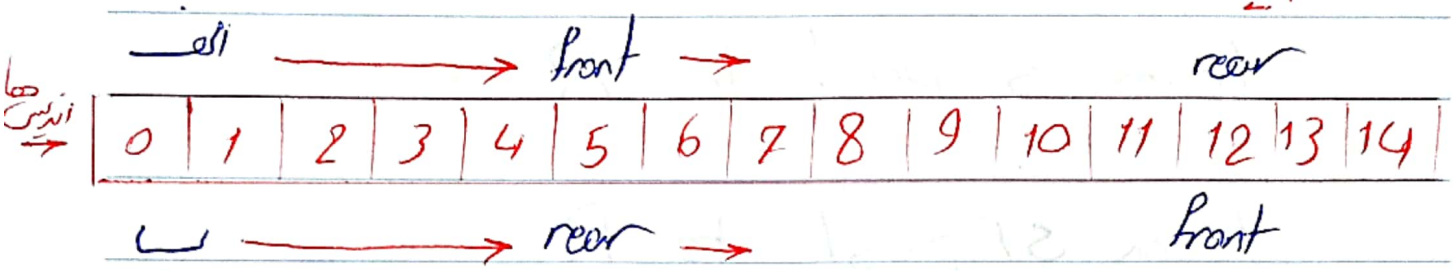
Date: / / 4011210 14 Sat Sun Mon Tue Thu Wed Fri Subject: لیلا محمدی حسنی

تمرین شماره 6

۱ـ



الف در قسمت الف front=5 هست که یعنی صف از خانه 6 شروع می‌شود و در خانه 12 به پایان می‌رسد. پس 7 عنصر دارد.

ب در قسمت ب front=12 هست یعنی ابتدای صف خانه 13 می‌باشد و در خانه 5 که مقدار rear هست به پایان می‌رسد. پس 8 عنصر دارد.

ج)

حالت کلی ← طول آرایه % (طول آرایه − rear + front)

حالت کلی
$$\begin{cases} r-f & rear > front \\ n-|r-f| & rear < front \end{cases}$$

Date: / / Sat Sun Mon Tue Thu Wed Fri Subject: ----------

سوال 2

1- pop 5 from S and push it to R

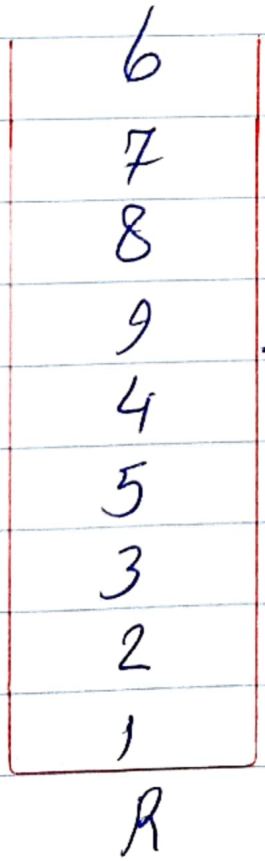
2- pop 4 from S and push it to R

3- pop 9 from T and push it to R

4- pop 8 from T and push it to R

5- pop 7 from T and push it to R

6- pop 6 from T and push it to R



تا اینجا R به شکل روبرو است و آوی خالی هستند.

Date: pop 6 Sat Sun Mon Tue Thu Wed Fri Subject: ----------

7 - r from R and push it to S

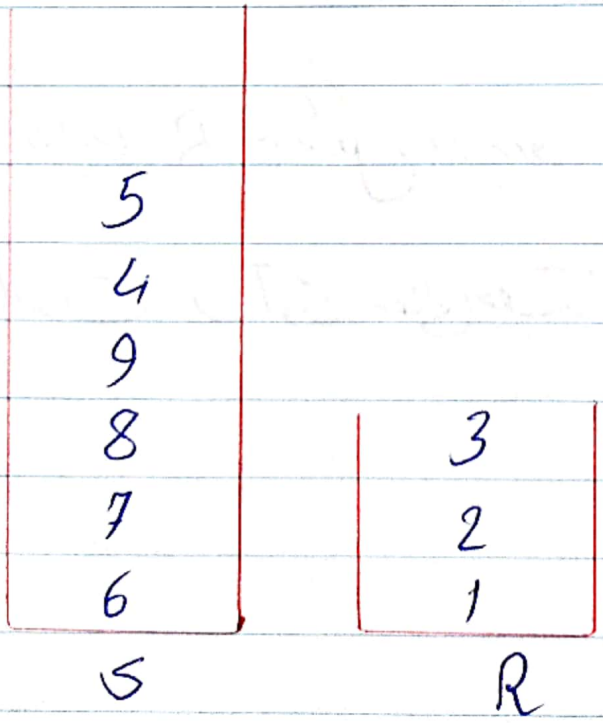
8 - pop 7 from R and " "

9 - pop 8 from R " "

10 - pop 9 " " "

11 - pop 4 " " "

12 - pop 5 " " "



Date: / / Sat Sun Mon Tue Thu Wed Fri Subject: ----------

```
internal class Node {
    internal int data;
    internal Node next;
    internal Node previous;
    public Node(int data){
        this.next = null;
        this.previous = null;
        this.data = data;
    } public Node DeepCopy(){
        Node node = (Node) this.MemberwiseClone();
        return node;
}}}
```

Date: / / Sat Sun Mon Tue Thu Wed Fri Subject:

```
internal class DoublyLinkedList {
    private Node? head;
    private Node? tail;
    public bool IsEmpty() => head is null;

    public void InsertAtLast(int data){
        Node node = new(data);
        InsertNode(node);
    }
```

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject:

```
private void InsertNode(Node node){
    if (IsEmpty()) {
        this.head = node;
    } else {
        this.tail!.next = node;
        node.previous = tail;
    }
    this.tail = node;
}
```

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject: ----------

```
public void showForward() {
    Node? node = head;
    while (node is not null) {
        Console.Write(node.data + "-->");
        node = node.next;
    }
    Console.WriteLine();
}
```

Date: / / Sat Sun Mon Tue Thu Wed Fri Subject: ----------

```
public void showBackward(){
    node? node = tail;
    while (node is not null){
        Console.Write(node.data + "--->");
        node = node.previous;
    }
    Console.WriteLine();
}

private bool IsNotEmpty() => !IsEmpty();
```

Date: / / Sat Sun Mon Tue Thu Wed Fri Subject: ----------

```
private void swapeReference (Node node){
    var temp = node.previous;
    node.previous = node.next;
    node.next = temp;
}
```

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject: ----------

```
public DoublyLinkedList ImmutableBackward(){
    DoublyLinkedList list = new();
    if( Is Not Empty){
    Node CurrentNode = this.tail;
    while( CurrentNode is not null){
     CurrentNode = currentNode.DeepCopy();
     var pre node = current node.previous;
     SwapeReference( currentNode);
     list.InsertNode(CurrentNode);
     CurrentNode = pre node;
    }} return list;}
```

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject: ----------

```
public void Reverse() {
    if (Is Not Empty) {
        Node? current Node = His.tail;
        var pre node = current Node.previous;
        swapeReference (current Node);
        Current Node = pre node;
    }
    var temp = tail;
    tail = head;
    head = temp;
}}}}
```

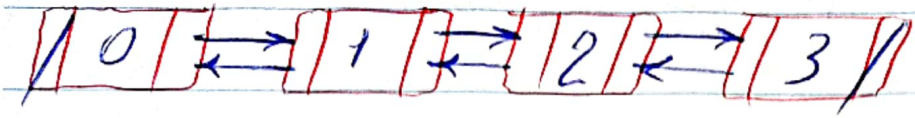
Date: / / Sat Sun Mon Tue Thu Wed Fri Subject: ----------

```
class Program {
    static void Main(string[] args){
DoublyLinkedList list = new DoublyLinkedList();
list.InsertAtLast(0); list.InsertAtLast(1);
list.InsertAtLast(2); list.InsertAtLast(3);
list.showForward();
list.showBackward();
DoublyLinkedList backwardList =
  list.ImmutableBackward();
backwardList.showForward();
backwardList.showBackward();
```

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject: ----------

```
list.Reverse();
list.showForward();
list.showBackward();}}}
```

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject: ----------



کلاس ها:

program : وظیفه اجرای برنامه و فراخوانی توابع

node : کلاسی برای ساختن node و گره ها.

پروپرتی ها:

- data از نوع int

- next از نوع node برای اشاره به گره بعد

- previous از نوع node برای اشاره به گره قبل

متد ها:

- Constructor : برای set کردن مقدار data

- Deep Copy : کپی گرفتن از obj با تغییر رفرنس

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri. Subject: ----------

DoublyLinkedList : برای ساختن لیست دوسویه

پروپرتی ها:

- head از نوع node برای اشاره به ابتدای لیست
- tail از نوع node برای اشاره به انتهای لیست

متد ها:

- IsEmpty : برای تست خالی بودن لیست
- InsertAtLast : برای اضافه کردن به انتهای لیست
- showBackward : برای نمایش برعکس لیست مال کسی که سوال است پیاده می شود.
- IsNotEmpty : برای تست خالی نبودن لیست
- swapeRefrence : برای تعویض next و prev یک نود

Date: / / Sat. Sun. Mon. Tue. Thu. Wed. Fri Subject:

- Reverse : تعریف next و prev همه نودها

بصورت ← Mutable. Ex (Mutable Backward)

ImmutableBackward : برعکس کردن لیست به صورت Immutable.

- Insert Node : برای اضافه نود به آخر لیست.