Java

```
class StacQ<T>
{
   private BinNode<T> start;      // תחילת השרשרת
   private BinNode<T> end;        // סוף השרשרת
   public StacQ(T x)              // פעולה בונה
   {
      this.start = new BinNode<T>(x);
      this.end = this.start;
   }

   public void add(int side, T x) // פעולה המוסיפה איבר לצד 1 או 2
   {
      BinNode<T> temp = new BinNode<T>(x);
      switch (side)
      {
         case 1: if (this.start == null) // אם זהו האיבר הראשון שהוכנס
                 {
                    this.start = temp;
                    this.end = this.start;
                 }
                 else
                 {
                    temp.setRight(this.start);
                    this.start.setLeft(temp);
                    this.start = temp;
                 }
                 break;
         case 2: if (this.end == null) // אם זהו האיבר הראשון שהוכנס
                 {
                    this.end = temp;
                    this.start = this.end;
                 }
                 else
                 {
                    temp.setLeft(this.end);
                    this.end.setRight(temp);
                    this.end = temp;
                 }
                 break;
      }
   }

   public T remove(int side) // פעולה המוציאה איבר מצד 1 או 2
   {
      if (!this.isEmpty())       // האם האוסף לא ריק
      {
         BinNode<T> temp;
         switch (side)
         {
```

סיבוכיות זמן הריצה של הפעולה StacQ: **O(1)**

סיבוכיות זמן הריצה של הפעולה add: **O(1)**

```
            case 1: temp = this.start;
                    this.start = this.start.getRight();
                    if (this.start == null)    // אם זהו האיבר האחרון
                       this.end = null;
                    else
                       this.start.setLeft(null);
                    temp.setRight(null);
                    return temp.getValue();
            case 2: temp = this.end;
                    this.end = this.end.getLeft();
                    if (this.end == null)     // אם זהו האיבר האחרון
                       this.start = null;
                    else
                       this.end.setRight(null);
                    temp.setLeft(null);
                    return temp.getValue();
          }
       }
       return null;
}
```

סיבוכיות זמן הריצה של הפעולה remove: O(1)

שימו לב! יש להתייחס גם למקרי קיצון בפעולות של הוספת והוצאת איבר. בפעולת הכנסת איבר יש להתייחס למקרה שזהו האיבר הראשון שמוכנס, ובפעולת הוצאת איבר יש להתייחס למקרה של הוצאת האיבר האחרון מהאוסף.

```
   public T getSide(int side)       // פעולה המחזירה את ערכו של האיבר הנמצא בצד 1 או 2
   {
      switch (side)
      {
         case 1: return this.start.getValue();
         case 2: return this.end.getValue();
      }
      return null;
   }

   public boolean isEmpty() // פעולה הבודקת אם האוסף ריק או לא
   {
      return (this.start == null && this.end == null);
   }
}
```

סיבוכיות זמן הריצה של הפעולה getSide(): O(1)

סיבוכיות זמן הריצה של הפעולה isEmpty: O(1)

סעיף ג

Java

```
public static int sideRezef(StacQ<Character> s)
{
   BinNode<Character> start = s.getStart(),end = s.getEnd();
```

```
if (!s.isEmpty())
{
        char prev1 = start.getValue(), prev2 = end.getValue();
        start = start.getRight();
        end = end.getLeft();
}
boolean ok1 = true, ok2 = true;
while (start != null && end != null && ok1 && ok2)
{
   if (start.getValue() != prev1)
          ok1 = false;
   if (end.getValue() != prev2)
          ok2 = false;
   start = start.getRight();
   end = end.getLeft();
}
if (ok2&&!ok1)
      return 2;
return 1;
```

