T.C
Sivas Cumhuriyet Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
2019-2020 Öğretim Yılı Veri Yapıları Dersi Ara Sınav Soruları

Öğrenci No : **18.11.2019**
Adı Soyadı :

SORULAR

1. Java Programlama dilinde dinamik yapıda ve kendine ait bir metot kütüphanesine sahip dizi yapılarına **ArrayList** denir. **ArrayList** sınıfı iki şekilde tanımlanabilir:

```
ArrayList<Integer> listName = new ArrayList<Integer>(int initialCapacity);
ArrayList<Integer> listName = new ArrayList<Integer>();
```

- **`initialCapacity`** verilen **`List`** yapısı bu boyuta göre oluşturulur,
- Boyut belirtilmeyen **`List`** yapısı ise default olarak **10** kapasite ile oluşturulur.
- Her iki durumda da diziye veri eklendiğinde dizi kapasitesinin aşılması durumu söz konusu ise arka planda dizi boyutu %50 artırılır. Her aşımda bu işlem tekrarlanır.

Buna göre; aşağıda verilen dizi sınıfında gerekli düzenlemeleri yaparak ArrayList'e ait yukarıda belirtilen özellikleri Array veri yapısı için sağlayınız. (15p)

```
class Array {
      private long[] dizi;
      private int elemanSayisi;

      public Array(int max){
            dizi = new long[max];
            elemanSayisi = 0;
      }

      public void add(int value)
      {
            dizi[elemanSayisi] = value;
            elemanSayisi++;
      }
```

2. Aşağıda tanımlanan Link yapısına uygun olacak şekilde başlığı verilen **`sublist`** metodunu tamamlayınız.(25p)

```
public void subList(int first, int last) {

      ...
      }
```

3. Aşağıda tanımlanan Link yapısına uygun olacak şekilde **`deleteAfter`** metodunu **recursive** bir metot olarak yeniden yazınız.(30p)

```
public boolean delAfter(Link head, int key) {
      ...
      }
```

4. Aşağıdaki kodun ekran çıktısı nedir? (10p)

```
class LinkListApp {
public static void main(String[] args) {
      LinkList theList = new LinkList();

      theList.insertToTail(11);
      theList.insertToTail(22);
      theList.insertToTail(33);
      System.out.println(theList.MethodX(theList.getHead()));
      }
}
```

5. Aşağıdaki koda ait ekran çıktısını yazınız.(5p)

```
public class Value {
	static int x = 5;
	public static void main(String[] args) {
		x = 10;
		int a = x;
		a= 1;
		System.out.println(x);
		x=5;
		System.out.println(a);
		int [] dizi1 = {1,2,3};
		int [] dizi2 = {1,2,3};
		int [] dizi3;

		dizi2[1]=5;
		dizi1=dizi2;
		System.out.println(dizi1[1]);
	}
}
```

6. 200 elemanlı bir dizide Binary Search algoritmasının tamamlanması için bakılması gereken maksimum eleman sayısı: (3p)

- **a.** 200
- **b.** 8
- **c.** 1
- **d.** 13 tür.

7. Sıralı diziler, karışık dizilerle karşılaştırıldıklarında; (3p)

- **a.** Silmede daha hızlıdır.
- **b.** Eklemede daha hızlıdır.
- **c.** Create etmede daha hızlıdır.
- **d.** Aramada daha hızlıdır.

8. Aşağıdakilerden hangisi doğrudur? (3p)

- **a.** Bir yığındaki pop işlemi, bir kuyruktaki çıkarma işleminden oldukça basittir.
- **b.** Bir kuyruğun içeriği, başa sarabilirken, bir yığının içeriğinde bu olamaz.
- **c.** Bir yığının top değeri, sıranın first değerine karşılık gelir.
- **d.** Hem yığın hem de kuyrukta, sırayla çıkartılan öğeler, dizideki gittikçe daha yüksek indeksli hücrelerden alınır.

9. Öncelik kuyruklarındaki (priority queue) öncelik (priority) terimi ne anlamda kullanılır? (3p)

- **a.** Önce en yüksek öncelikli öğeler eklenir.
- **b.** Programcının, ele alınan diziye erişime öncelik vermesi gerekir.
- **c.** Ele alınan dizi, öğelerin önceliğine göre sıralanır.
- **d.** Önce en düşük öncelikli öğeler silinir.

10. 4. soruda programın son satırına gelindiğinde dizi1, dizi2 ve dizi3 dizilerinin Heap bellekte kapladıkları toplan alan ne kadardır? (3p)

- **a.** 36 byte
- **b.** 24 byte
- **c.** 12 byte
- **d.** Yer kaplamaz.

```
class Link {
   public int iData;
   public Link next;

   public Link(int id, double dd) {
      iData = id;
      next = null;
       }
    }//end class
//--------------------------------------------
class LinkList {
   private Link head;
   private Link tail;

   public LinkList() {
         head = null;
         tail = null;
    }
    public Link find(Link head, int key) {

         if (head == null)
               return null;
         if (head.iData == key)
               return head;
         return find(head.next, key);
         }
    public boolean deleteAfter(int key){
         Link current = head;
         while (current.iData != key)
         {
               if (current.next == null)
                     return false;
               else
                     current = current.next;
         }

         if (current.next != null) {
               current.next = current.next.next;
               return true;
         }
         else
               return false;
         }

    public void insertToTail(int id) {
         Link newLink = new Link(id);
         if (isEmpty())
               head = newLink;
         else
               tail.next = newLink;
         tail = newLink;
         }

    public int MethodX(Link head) {
         if (head == null)
               return 0;

         return 1+MethodX(head.next);
         }

    public Link getHead() {
         return this.head;
         }
    }//end class
```