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

Öğrenci No : **02.01.2020**
Adı Soyadı :

SORULAR

1. Hash Table veri yapısının boyutunun artırılması gerektiğinde aşağıdaki kuralları sağlaması istenmektedir:

- Dizideki veri miktarı, dizi kapasitesinin 2/3'ünü aştığında genişletme yapılacaktır.
- Yeni dizinin kapasitesi, eski dizinin kapasitesinin 2 katına denk gelen sayıdan sonraki ilk asal sayı olacaktır.
- *(**Asal sayı kontrolü ayrı metot içerisinde yapılacaktır)*
- Eski dizideki veriler yeni diziye rehashing ile aktarılacaktır.
- Aktarım işlemi yapılırken `null` ve `-1` değerler alınmayacaktır.
- Kodlarınızı **Ek-1**'de verilen metotlara göre ve boş bırakılan metotları (boyutkontrol, asalGetir, asalMi) doldurarak yazmanız gerekmektedir. Fazladan herhangi bir metot eklentisi kabul edilmeyecektir.

Yukarıdaki şartlar sağlandığında Hash Table yapısını büyüten kodu yazınız.(25)

2. Ek-2'de tanımlanan Link yapısına uygun olacak şekilde **`aralikSil`** metodunu yazınız.(first ve last değerleri listedeki linklerin sıra numarasını ifade etmektedir)(25p)

```
public boolean aralikSil(int first, int last)
    {
          ...
    }
```

3. Ek-2'deki tanımlı class' ları kullanan aşağıdaki kodun ekran çıktısını yazınız ve kısaca anlatınız? (15p)

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

    theList.insertToTail(1);
    theList.insertToTail(5);
    theList.insertToTail(4);
    theList.insertToTail(3);
    theList.insertToTail(2);
    for (int i = 0; i < 5; i++) {
     if (theList.methodX() != null) {
       System.out.print(theList.methodX().iData
       + " ");
       theList.delete(theList.methodX().iData);
     }
    }
  } // end main()
}// end class
```

3. Soru
Çıktı:
Açıklama:

4. Ek-2'deki tanımlı class' ları kullanan aşağıdaki kodun ekran çıktısını yazınız ve kısaca anlatınız? (14p)

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

     theList.insertToTail(1);
     theList.insertToTail(5);
     theList.insertToTail(4);
     theList.insertToTail(3);
     theList.insertToTail(2);

      System.out.println(theList.methodY(theList
.getHead()));
  } // end main()
}
```

4. Soru
Çıktı:
Açıklama:

5. MergeSort algoritmasının dezavantajı nedir?(3p)

a. Recursive değildir.

b. Daha fazla bellek kullanır.

c. InsertionSort' dan daha hızlı iken, quickSort'a göre daha yavaştır.

d. Uygulaması karmaşıktır.

6. Sırasız bir diziye eleman eklemek; (3p)

a. Dizinin boyutuyla doğru orantılı zaman alır.

b. Çoklu karşılaştırma gerektirir.

c. Alan oluşturmak için kaydırma gerektirir.

d. Ne kadar eleman olduğuna bakmaksızın aynı zamanı alır.

7. Bir dengesiz (unbalanced) ağaç yapısında; (3p)

a. Anahtar değerlerin çoğu ortalamadan büyüktür,

b. Davranışı tahmin edilemez,

c. Kök veya başka bir düğümde sağ çocuklardan daha fazla sol çocuğu vardır (ya da tersi),

d. Bir şemsiye şeklindedir.

8. Ağaç veri yapısının bir alt-ağacı (subtree) her zaman; (3p)

a. Ana ağacın kökünün çocuğu olan köke sahiptir.

b. Ana ağacın köküne bağlı olmayan bir köke sahiptir.

c. Ana ağaçtan daha az düğüme sahiptir.

d. Aynı sayıda düğüme sahip bir kardeşe sahiptir.

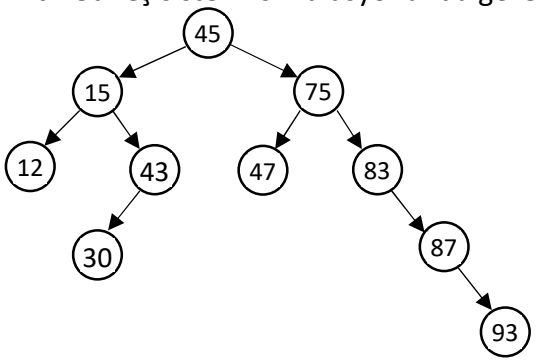
9. Bir öncelik kuyruğu ile aşağıdakilerin hangisi tutulabilir?(3p)

a. Şehri farklı yerlerinden gelen taksi çağrıları.

b. Bilgisayar klavyesinden yapılan tuş vuruşları.

c. Bir oyundaki satranç tahtasının kareleri

d. Güneş sistemi simülasyonunda gezegenler.



10. Yukarıda verilen binary search tree için PostOrder dolaşım yapıldığında aşağıdakilerden hangisi çıktı olark elde edilir?(3p)

a. 45, 15, 12, 43, 30, 75, 47, 83, 87, 93

b. 12, 30, 43, 15, 47, 93, 87, 83, 75, 45

c. 93, 87, 83, 75, 47, 45, 43, 30, 15, 12

d. 87, 15, 12, 30, 43, 45, 83, 97, 75, 93

11. Ağaçtaki sayıların küçükten büyüğe sıralı şekilde çıktısını elde edebilmek için hangi dolaşım (traverse) metodu kullanılır?(3p)

a. PreOrder **c.** PostOrder

b. InOrder **d.** AscOrder

	a	b	c	d
5.	O	O	O	O
6.	O	O	O	O
7.	O	O	O	O
8.	O	O	O	O
9.	O	O	O	O
10.	O	O	O	O
11.	O	O	O	O

Cevap-2:

Ek-1

```
public class DataItem {
   private int iData;

   public DataItem(int ii)
   {
      iData = ii;
   }
}
//--------------------------------------
public class HashTable {
   private DataItem[] hashArray;
   private int arraySize;
   private int elemanSayisi;

   public HashTable(int size)
   {
      arraySize = size;
      hashArray = new DataItem[arraySize];
      elemanSayisi = 0;
   }
   public int hashFunc(int key) {
      return key % arraySize;
   }

public void insert(DataItem item){
   boyutKontrol();
//Kontrol yapılacak ve ihtiyaç varsa büyütülecek.
   int key = item.getKey();
   int hashVal = hashFunc(key);

   while (hashArray[hashVal] != null &&
         hashArray[hashVal].getKey() != -1)
   {
      ++hashVal;
      hashVal %= arraySize;
   }
   hashArray[hashVal] = item;
   elemanSayisi++;
}

public void boyutKontrol() {
   ……
}
private int asalGetir(int min){
   ……
}
private boolean asalMi(int n){
   ……
}
}//class HashTable
```

Ek-2

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

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

   public LinkList() {
      head = null;
      tail = null;
   }
//--------------------------------------
   public boolean isEmpty(){
      return (head == null);
   }
//--------------------------------------
   public Link methodX() {
      if (head == null)
         return null;
      else {
         Link current = head.next;
         Link myLink = head;
         while (current != null) {
            if(current.iData > myLink.iData)
               myLink = current;
            current = current.next;
         }
         return myLink;
      }
   }// methodX
//--------------------------------------
   public void delete(int data) {
    if (!isEmpty())
      if (head == tail && data == head.iData)
         head = tail = null;
      else if (data == head.iData)
         head = head.next;
      else {
         Link previous;
         Link current;
         previous = head;
         current = head.next;
         while(current!=null &&
                   current.iData!=data) {
            previous = previous.next;
            current = current.next;
         }//while
         if (current != null) {
            previous.next = current.next;
            if (current == tail)
                  tail = previous;
         }//if
      }//else
   }//delete
//--------------------------------------
   public void insertToTail(int data) {
      Link newLink = new Link(data);
      if (isEmpty())
         head = newLink;
      else
         tail.next = newLink;
         tail = newLink;
   }//insertToTail
//--------------------------------------
   public int methodY (Link head) {
      if (head == null)
         return 0;
      int x = head.iData*head.iData;
      return x + methodY(head.next);
   }//methodY
//--------------------------------------
   public Link getHead() {
      return this.head;
   }
}//end class
```