BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BIL203 Programlama - II

Ara sınav – Aralık 2010

Ad Soyad	
Numara	
İmza	

1	2	3	4	5	6	7	8	9	Toplam

Açıklamalar:

1. Toplam soru adedi ..... dir.
2. Cevaplama süresi ..... dakikadır.
3. Gerçeklemelerinizi verilen boşluklara yazınız.

1. (23 p) Yığıt

Operator olarak sadece `VE` ve `VEYA` ifadelerinin yer aldığı mantıksal ifadeler için postfix dönüşümü yapan `bol_in2postfix` işlevini gerçekleyiniz. Mantıksal ifadede değişkenler ve işleçler boşlukla ayrılmaktadır. İfadelerde yer alan "`A B`", "`A VE B`" anlamına; "`C + D`" ise "`C VEYA D`" anlamına gelmektedir. Postfix gösterilimi sunulurken `VE` için "`*`"; `VEYA` için "`+`" kullanılacaktır.

```
def  bol_in2postfix(bexpr):
     """\
     >>> bol_in2postfix("A B C + D")
     A B * C * D +
     """
     tmp = bexpr.split()
     token_list = []
     state = 0

     for token in tmp:
          if token in str.uppercase and state == 1:
               token_list.append('*')
          token_list.append(token)

          if token in str.uppercase:
               state = 1
          else:
               state = 0
     # gerisi infix2postfix
     ...
```

2.(18 p) Yığıt

Sürekli ikiye bölme temelinde onluk sayı tabanından ikilik sayı tabanına dönüşüm yapan `dec2bin` (veya `divideBy2`) işlevine benzer, yığıtı etkin kullanan, sürekli ikiyle çarpma temelinde ikilik tabandan onluk tabana dönüştürme yapan `bin2dec` işlevini gerçekleyin.

İpucu: bins = 1011 için, dec=0 ile başla;

b=1 için (en anlamlı bit)	dec = 2*dec + b (dec=1);
b=0 için	dec = 2*dec + b (dec=2);
b=1 için	dec = 2*dec + b (dec=5);
b=1 için (en az anlamlı bit)	dec = 2*dec + b (dec=11)

```
def bin2dec(bins):
    """\
    >>> bin2dec("1011")
    11
    """
    bStack = stack()
    for b in bins:
        bStack.push(b)

    dec = 0
    while not bStack.isEmpty():
        dec = 2 * dec + int(bStack.pop())

    return dec
```

3. (10 p) Yığıt

Aşağıdaki PostScript kodunun üreteceği sonucu (işlem sonucunda yığıtta bulunan değer(ler)i hesaplayınız. Sonuca ulaşmak için yaptığınız işlemleri adım adım açıklayınız.

6 2 sub 4 3 add mul 18 sub 10 div 10 sub

Yığıt ile çözüme gidilebilir. Yığıtın içeriği her bir adımda aşağıdaki gibi olacaktır ('yukarı = sol' kabul edersek): [] --> [6] --> [2 6] --> [4] --> [4 4] --> [3 4 4] --> [7 4] --> [28] --> [18 28] --> [10] --> [10 10] --> [1] --> [10 1] --> [-9]	Bu PostScript kodu önce postfix' e çevrilir: 6 2 - 4 3 + * 18 - 10 / 10 - Daha sonra infix' e çevrilir ve sonuç hesaplanır: ((6 - 2) * (4 + 3) - 18) / 10 - 10 = -9

4. (22 p) Kuyruk, SVT

Kuyruk (`Queue`) SVT (soyut veri türü) gerçeklemesinden miraslanan, `N` elemanla sınırlanan, halka kuyruk SVT'nı gerçekleyin.

İpucu: aşağıda verilen örnek çalışma yorum kısmındaki indislerin anlamı: `e` den çek, `d` nin ardına ekle anlamında kullanılmaktadır.

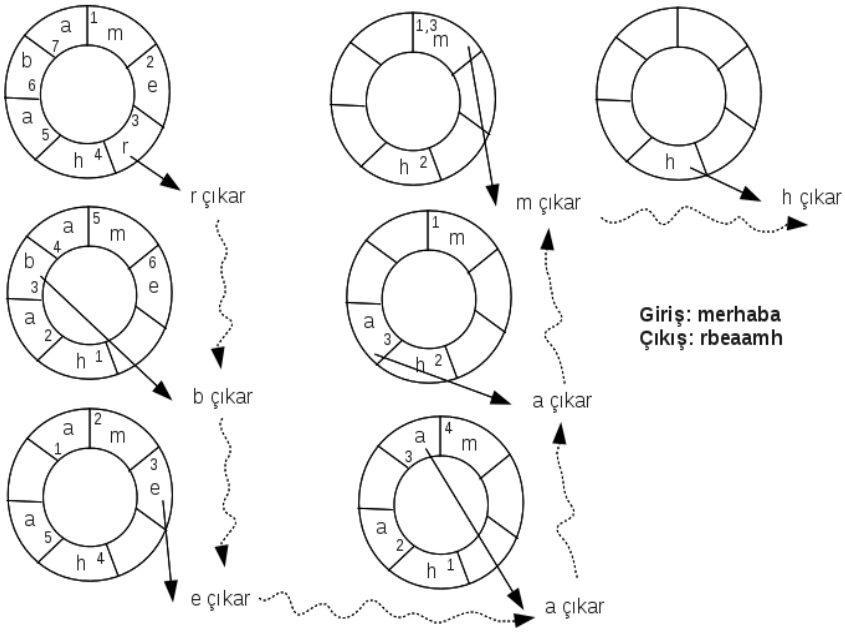
```
class HalkaKuyruk(Queue):
    """\
    >>> hk = HalkaKuyruk(3)
    >>> hk.enq("A")          # [    A];          e=1, d=1
    >>> hk.enq("B")          # [  B A];          e=1, d=2
    >>> hk.enq("C")          # [C B A];          e=1, d=3
    >>> hk.enq("D")          # [C B D];          e=2, d=1
    >>> hk.deq()             # [C B D];          e=3, d=1
    B
    """
    def __init__(self, N):
        self.N = N
        self.items = [None] * N
        self.e = 1
        self.d = 0

    def enq(self, item):
        self.items[self.d + 1] = item
        self.d = (self.d + 1) % self.N

    def deq(self):
        t = self.items[self.e]
        self.e = (self.e + 1) % self.N + 1
        return t

    def isEmpty(self):
        return Queue.isEmpty(self)

    def size(self):
        return Queue.size(self)
```

5. (28 p) Kuyruk

Algoritması takip eden biçimde olan bir kelime şifreleyici gerçeklemeniz istenmektedir. Giriş olarak `dizgi="merhaba"` verildiğinde ve döndürme miktarı `N=3` seçildiğinde ara aşamalar sağ taraftaki çizimde verilmiştir.

Algoritma:

1. Kelime (`dizgi`) içerisindeki harfleri sırayla disk üzerine yerleştir
2. diski belli miktar (`N`) döndür
3. denk gelen karakteri diskten çıkar
4. çıkan karakterleri çıkış listesinde birleştir
5. disk üzerinde karakter kalmayıncaya kadar 2-3-4 adımlarını tekrarla
6. çıkış listesini dizgiye dönüştür




```
def sifrele(dizgi, N):
    """\
    >>> sifrele("merhaba", 3)
    rbeaamh
    """
    disk = Queue()
    for h in dizgi:
        disk.enq(h)

    rlist = []
    while disk.size() > 1:
        for i in range(N - 1):
            disk.enq(disk.deq())
        rlist.append(disk.deq())

    rlist.append(disk.deq())

    return join(rlist, " ")
```

Yardımcı işlevler

```
class Stack():
      def __init__(self):
            self.items=[]

      def isEmpty(self):
            return self.items==[]

      def push(self,item):
            self.items.append(item)

      def pop(self):
            return self.items.pop()


      def peek(self):
            return self.items[len(self.items)-1]

      def size(self):
            return len(self.items)
```

```
def divideBy2(decNumber): # dec2bin

    remstack = Stack()

    while decNumber > 0:
            rem = decNumber % 2
            remstack.push(rem)
            decNumber = decNumber / 2

    binString = ""
    while not remstack.isEmpty():
            binString = binString + repr(remstack.pop())

    return binString
```

```
class Queue:
    def __init__(self):
             self.items = []

    def isEmpty(self):
             return self.items == []

    def enqueue(self, item):
             self.items.insert(0,item)

    def dequeue(self):
             return self.items.pop()

    def size(self):
             return len(self.items)
```