Ad Soyad		İmza	
Numara			
Tarih	09/12/09		

ONDOKUZ MAYIS ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİL205 Veri Yapıları

ARASINAV

1	2	3	4	5	6	7	8	9	10	Toplam
10	15	15	25	35						

SORULAR

1. (10 p) Tanım

a) Soyutlama ne demektir? **Arayüz, İstemci** ve **Gerçekleme** anahtar kelimeleri yardımıyla açıklayınız.

Soyutlama, **istemci**nin kendisine **arayüz** yardımıyla sunulan hizmetlerin **gerçekleme** ayrıntılarını gizleme yaklaşımıdır.

b) Veri Yapısı, Veri Türü ve Veri Modeli kavramlarının benzerlik ve ayrımlarını kısaca yazınız. Takip eden kavramlar hangi tanımla ilişkilendirilebilir: **int**, **struct** ve **tree** (ağaç).

Veri yapısı	Veri Türü	Veri Modeli
Saklanma biçimi	Veriyi açıklamak	İlişkisel düzen

int	struct	tree
Veri türü	Yeni veri modeli/türü	Veri modeli

c) LIFO ve FIFO kavramlarıyla yığıt ve kuyruk arasında nasıl bir ilişki vardır? Kısaca açıklayınız.

LIFO: yığıtın terssel özelliği

FIFO: kuyruğun ilk gelene ilk servis

2. (15 p) Python Programlama

a) Aşağıdaki kod çalıştırıldığında `letterlist` değişkeninin içeriğinde ne vardır?

```
wordlist = [’cat’, ’dog’, ’rabbit’]
letterlist = []
for aword in wordlist:
    for aletter in aword:
        letterlist.append(aletter)
```

`letterlist = ['c','a','t,'d','o','g','r','a','b','b','i','t']`

b) Aşağıdaki kod parçası çalıştırıldıktan sonra `y` değişkeni `2` değerine sahip oluyorsa `x` değişkeni hangi değerdedir?

```
if x > 3:
  if x <= 5:
    y = 1
  elif x != 6:
    y = 2
  else:
    y = 3
else:
  y = 4
```

`X ve y TAMSAYI. y = 2` ise `x > 6`

3. (15 p) İfade Çevrimi

`y = a b + c *` *Postfix* ifadesi için:

a) *Infix* ifade karşılığını yazınız.

Infix: (a + b) * c

b) *Prefix* ifade karşılığını yazınız.

Prefix: * + a b c

c) `a = 3, b = 4 ve c = 5` ise `y` nedir?

Y = (3 + 4) * 5 = 35

4. (25 p) Yığıt Soyut Veri Yapısı

Yığıt soyut veri yapısı gerçeklemesi aşağıda verildiğine göre:

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

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

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

    def pop(self):
    return self.items.pop(0)

    def peek(self):
    return self.items[0]

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

def chkExpr(expr):

    # kodlarınız buraya (4.b)
```

a) `s` ve `t` yığıtları başlangıçta boş ve `a, b, c ve d` python nesneleri olsun. Aşağıdaki işlemlerin sonucunda `s` ve `t` yığıt içerikleri ne olur?

```
s = Stack()
t = Stack()

s.push(a)
s.push(b)
s.push(c)
t.push(d)
t.push(s.pop())
t.push(s.peek())
s.push(t.pop())
t.pop()
```

```
b
b
a   d
=   =
s   t
```

Burada `peek()` yönteminin yığıtı güncellemeksizin tepedeki değeri döndürdüğünü hatırlamak gerekir.

s: a b b

t: d

b) *Infix* gösteriminde verilen ifadedeki parantezleri denetleyen `chkExpr(expr)` işlevini gerçekleyiniz. İki parantez türü mevcuttur: `()` ve `[]`.

Aşağıdaki iki örnekte gösterildiği gibi `chkExpr(expr)` işlevi parantez uyuşmazlığı olduğu durumda `False`, diğer durumda `True` değeri döndürecektir:

```
Doğru : [ ( 1 + 3 ) / ( 4 + 5 ) ]
Yanlış: [ ( 1 + 3 ) / ( 4 + 5 ]

Doğru : ( ( 1 + 2 + 3 ) / 4 )
Yanlış: ( ( 1 + 2 + 3 ) / 4
```

Gerçeklemeniz: genelleştirilmiş parantez denetçi kodu (listing 2.4) veya aşağıdaki kod satırları

```
def chkExpr(expr):

        opens = '[('

        close = '])'

        tokenList = expr.split()

        parStack = Stack()



        for token in tokenList:

                if token in opens:

                        parStack.push(token)

                elif token in close:

                        if parStack.isEmpty():

                                return False

                        ind = opens.index(parStack.pop())

                        if token != close[ind]:

                                return False

        if parStack.isEmpty():

                return True

        else:

                return False
```

5. (35 p) Kuyruk Soyut Veri Yapısı

Kuyruk soyut veri yapısı gerçeklemesi aşağıda verildiğine göre:

```
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)

    def splice(self, ek):
        # kodlarınız buraya (5.b)

    def ilk_eklenen(self):
        # kodlarınız buraya (5.c)
```

a) Kuyruk üzerinde aşağıdaki işlemler yapıldığında ekran çıktısı nasıl olur?

```
q = Queue()
q.enqueue(5)
q.enqueue(4)
q.enqueue(3)
q.enqueue(2)
q.enqueue(q.dequeue())
print q.dequeue(), q.dequeue()
```

Ekran Çıktısı	4 3

b) Kuyruğun sonuna `ek` kuyruğu ekleyen `splice(self, ek)` işlevini gerçekleyin.

Örnek kullanım aşağıda verilmiştir:

```
q = Queue()
q.enqueue(5), q.enqueue(4), q.enqueue(3), q.enqueue(2)
# q - kuyruk icerigi: [2 3 4 5]
ek = Queue()
ek.enqueue(a), ek.enqueue(b), ek.enqueue(c)
# ek - kuyruk icerigi: [c b a]
q.splice(ek)
# q - kuyruk icerigi: [c b a 2 3 4 5]
```

Gerçeklemeniz:

```
def splice(self, ek):
    while not ek.isEmpty():
        self.enqueue(ek.dequeue())
```

c) Kuyruk içeriğini güncellemeden (yığıttaki `peek(self)` işlevine benzer olarak) kuyruğa **ilk** eklenen değeri döndüren `ilk_eklenen(self)` işlevini gerçekleyiniz.

Örnek kullanım:

```
q = Queue()
q.enqueue(5)
q.enqueue(4)
q.enqueue(3)
q.enqueue(2)
print q.ilk_eklenen()
# ekranda "5" gorunecek, "q" kuyruk icerigi degismeyecek
```

Gerçeklemeniz:

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

Sınav süresi 90 dakikadır.... Başarılar....