BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ	Ad Soyad	
BIL203 Programlama - II	Numara	
Dönemsonu Sınavı – Ocak 2011	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. (14 p) Dizi, SVT

En son gelene ilk hizmet verme temelli yığıt soyut veri türü bir çok algoritmanın yapı taşıdır. Dizi kullanarak yığıta ait push() ve pop() işlevlerini gerçekleyin. Gerçeklemeniz yığıtın boş (çekemezsin) ve dolu (daha eleman alamaz) durumunu idare edebilmelidir.

```
#define MAXVAL 100       /* val y□□□ t□n□n max. derinli□i */
int sp = 0;                   /* bir sonraki bo□ y□□□t konumu */
double val[MAXVAL];      /* de□er y□□□ t□ */

/* push: f de□erini y□□□ta it */
void push(double f) {




}

/* pop: y□□□ t□n en üstündeki de□eri çek */
double pop(void) {




}
```

2..(19 p) dizgi

a) s2 dizgisini, s1 dizgisinin ardına boşlukla ekleyen strscat(s1, s2) işlevini gerçekleyin. Ör. strscat(“merhaba”, “dunya”) -> “merhaba dunya”.

```
void strscat(char *s1, char *s2) {




}
```

b) s2 dizgisi, s1 dizgisinin sonunda bulunuyorsa 1, diğer durumda 0 değeri döndüren strend(s1, s2) isimli işlevi gerçekleyin. Ör. strend(“merhaba”, “ab”) -> “1” üretirken, strend(“merhaba”, “kaba”) -> “0” değeri üretmelidir.

```
int strend(char *s1, char *s2) {




}
```

3. (24 p) yapı, modülerite

```
typedef struct _Nokta {
     int x;
     int y;
} Nokta;
```

a) Yukarıdaki **Nokta** yapı tanımlamasını baz alarak.

```
/* makepoint: x ve y bile enlerinden bir nokta olu turur */
Nokta makepoint(int x, int y) {




}
/* addpoint: iki noktay  topla */
Nokta addpoint(Nokta n1, Nokta n2) {




}
```

b) Nokta yapısından yararlanılarak **Dikdortgen** yapısını gerçekleyin.

```
/* Dikdortgen: iki Noktal dikdortgen temsili */
typedef struct _ Dikdortgen {
     ?              nSU;          /* sol-ust kose noktasi */
     ?              nSA;          /* sag-alt kose noktasi */
} Dikdortgen;
```

c) Dikdortgen ve Nokta yapısını kullanarak verilen noktanın dikdörtgenin içerisindeyse “1”, diğer durumda “0” üreten nokta_iceridemi işlevini gerçekleyin.

```
int nokta_iceridemi(Dikdortgen d, Nokta n) {



}
```

4. (30 p) özyineleme, SVT, bağlı liste

```
typedef struct _Dugum {
     char *kelime;
     struct _Dugum *sol;      /* sol cocuk */
     struct _Dugum *sag;      /* sag cocuk */
} Dugum;
Dugum *kok;
```

Ağaç yapısı programlama yapılarında sıklıkla kullanılan soyut veri türlerindendir (SVT). İkil ağaç maksimum iki çocuğu olan özel ağaçtır. Ağaç ise özyineli bir tarifle verilir: “ağacın her bir düğümü, kökü ve/veya sol/sağ çocuklu ağaçtır”. Yani her bir düğümde bir değer (burada “kelime”), sol ve sağında başka bir ağaç uzanır. Bu amaçla ağaçtaki her bir düğümü temsilen Dugum SVT verilmiştir. Buna göre,

a)

```
/* dugum_ekle: hangi çocuk bossa o tarafa dugum ekler. Her ikisi
de doluysa hata verir. */
Dugum dugum_ekle(Dugum *d, char *k) {



}
```

b) Özyineleme

```
/* dugum_yazdir: d dugum listesini sirali bir sekilde yazdir. *
 *  Ozyineli olarak once sol, sonra dugum degeri, en son sag */
void dugum_yazdir(Dugum d) {









}
```

5. (13 p)dizgi

Yılın kaçıncı gününde olduğumuzu söyleyen yilgun işlevini gerçekleyin. Ör. yil_gun(“21.10.2010”) -> 294 üretmelidir. İşlev girdisi dizgi, günün ise “gün.ay.yıl” formatında girildiğine ve 4 yılda bir şubatta ortaya çıkan değişime dikkat ederek gerçeklemenizi yapınız.

```
/* yil_gun: gun, yilin kacinci gunudur */
int yil_gun(char *gun) {









}
```