

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BLM0111 – Algoritmalar ve Programlama
Bütünleme Sınavı

Ad&Soyad	:	
Öğrenci Numarası	:	

Akademik yıl : 2021-2022
Dönem : Güz
Tarih : 9 Şubat 2022 – 13:30
Sınav süresi : 110 dakika
Öğr. görevlisi : Dr. Öğr. Üyesi Ergün GÜMÜŞ

Soru	1	2	3	4	5	Toplam
Puan	20	15	20	20	25	100
Not						

KURALLAR

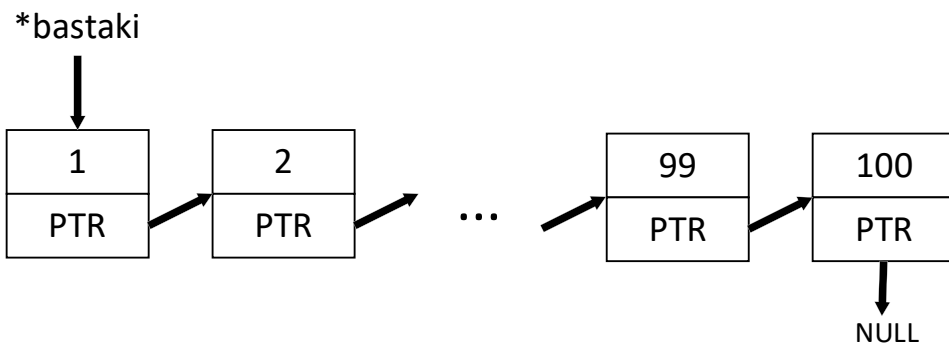
- Sınava başlamadan önce Ad&Soyad ve Öğrenci numarası alanlarını doldurunuz.
- Sınav öncesinde ve süresince sınav gözetmenlerinin tüm uyarılarına uymanız gerekmektedir.
- Sınav öncesinde cep telefonlarınızı KAPATINIZ!
- Soruları yanıtlamak için sadece sınav kâğıdınızla beraber verilen kâğıtları kullanmanız gerekmektedir. Yanıtlarınız açık ve okunaklı olmalıdır.
- Sınav boyunca masanızın üzerinde bulunabilecek malzemeler sadece sınav kâğıdınız, kalem ve silgidir.
- Sınav süresince herhangi bir nedenle birbirinizle konuşmak ve malzeme (silgi, kalem, kâğıt vb.) alışverişi yasaktır.
- Bu kuralların herhangi birine uymamak kopya çekmeye yönelik bir hareket olarak değerlendirilir ve ilgili makamlara bildirilir.

Sorular

1) [20p] Aşağıda “Dugum” isimli bir yapıya ait tanımlama görülmektedir. Dugum yapısı hem bir tamsayı hem de kendisiyle aynı türde başka bir düğümün adresini işaret eden bir işaretçi barındırmaktadır.

```
struct D{
    int sayi;
    struct D *PTR;
};
typedef struct D Dugum;
```

Dugum yapısını kullanarak aşağıda şekli görülen 100 elemanlı zinciri oluşturmanız isteniyor.



Bu zincirdeki her bir eleman dinamik olarak oluşturulacak, içerisindeki “sayı” alanı doldurulacak ve kendisinden önceki elemanla “PTR” işaretçisi vasıtasıyla ilişkilendirilecektir. Bu işlem sırasında hiçbir şekilde dizi yapısı kullanılmayacaktır.

```
void main() {
    int i;
    Dugum *bastaki, *ondeki, *arkadaki;
```

a) [5p] Zincirin en başındaki düğümü dinamik olarak yaratınız. Bu düğümün *sayi* alanını 1 sayısı ile doldurup **bastaki* isimli işaretçinin bu düğümü işaret etmesini sağlayınız.

```
bastaki = (Dugum *) malloc (sizeof(Dugum));
bastaki->sayi = 1;
```

b) [15p] Bir döngü içerisinde, zincirin kalan 99 elemanının her birini dinamik olarak yaratacak, içlerindeki *sayi* alanını sıradaki sayıyla dolduracak ve **ondeki*, **arkadaki* işaretçilerinden faydalanarak ardışık düğümleri birbirine bağlayacak bir kod yazınız. En sondaki düğümün *PTR* alanının NULL değerine sahip olduğuna dikkat ediniz.

```
ondeki = bastaki;  
  
for(i = 2; i <= 100; i++) {  
    arkadaki = (Dugum *)malloc(sizeof(Dugum));  
    arkadaki->sayi = i;  
    ondeki->PTR = arkadaki;  
    ondeki = arkadaki;  
}  
arkadaki->PTR = NULL;
```

2) [15p] Aşağıdaki kodda görülen ternary ifadenin eşleniğini sadece switch-case yapısı kullanarak yazınız. Gerekirse cevap alanını iki sütuna bölünüz.

```
int a, b, c, d = 0;  
scanf("%d %d %d", &a, &b, &c);  
d = ( c >= 1 && c <= 3 ) ? 30 : ( a == 4 ) ? 40 : ( a == 5 ) ? ( b == 6 ) ? 60 : 70 : 80 ;
```

```
switch(c){  
    case 1:  
        ;  
    case 2:  
        ;  
    case 3:  
        d = 30;  
        break;  
    default:  
        switch(a){  
            case 4:  
                d = 40;  
                break;  
            case 5:  
                switch(b){  
                    case 6:  
                        d = 60;  
                        break;  
                    default:  
                        d = 70;  
                }  
                break;  
            default:  
                d = 80;  
        }  
    }  
}
```

3) [20p] Aşağıdaki kod parçasında global değişken olarak tanımlanan N değişkeni $N \times N$ boyutlu bir karesel matrisin satır/sütun sayısını ifade etmektedir. Kodun main() bloğunda $N \times N$ boyutlu bir matris oluşturulmakta ve matrisin içi rasgele sayılarla doldurulmaktadır. Ardından bu matris ust(...) isimli bir özyinelemeli fonksiyona argüman olarak aktarılarak matrisin üst üçgenindeki sayıların toplamı hesaplanıp geri döndürülmektedir. Buna göre ust(...) fonksiyonunu doldurunuz.

Not1: Herhangi bir döngü yapısı tanımlamanız durumunda sorudan puan alamazsınız!

Not2: Fonksiyonun imzasını değiştirmeye çalışırsanız sorudan puan alamazsınız!

Not3: Koda herhangi bir global değişken eklemeye çalışırsanız sorudan puan alamazsınız!

```
int N = 5; // matrisin satır veya sütun sayısı

int ust(int M[][N], int satir, int sutun) {

    if(satir == 0 && sutun == 1)
        return M[satir][sutun];
    else if(sutun == satir)
        return ust(M, satir - 1, N - 1);
    else
        return ust(M, satir, sutun - 1) +
M[satir][sutun];

}

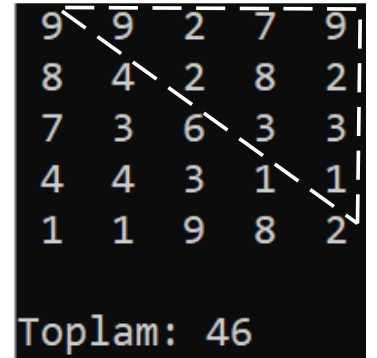
void main() {
    srand(time(NULL));
    int matris[N][N], i, j;

    for(i = 0; i < N; i++) {
        for(j = 0; j < N; j++) {
            matris[i][j] = rand() % 10 + 1;
            printf("%2d ", matris[i][j]);
        }
        printf("\n");
    }

    i = ust(matris, N - 2, N - 1);

    printf("\nToplam: %d", i);
}
```

Kodun örnek ekran çıktısı şu şekildedir:



```
9 9 2 7 9
8 4 2 8 2
7 3 6 3 3
4 4 3 1 1
1 1 9 8 2
Toplam: 46
```

4) [20p] Ali, “permütasyon şifrelemesi” adında bir teknik kullanılarak şifrelenip “sifreli.txt” dosyasına yazdırılmış bir metnin şifresini çözecek bir C programı yazmak istiyor. “sifreli.txt” dosyasının örnek içeriği yanda görülmektedir.

Dosya, iki sütun halinde yazılmış ve birbirlerinden TAB karakteri ile ayrılmış sayı & harf çiftlerini içermektedir. Her sayı, yanındaki harfin şifresiz metnin kaçınıcı harfi olduğunu gösteren bir indistir. Örneğin, ilk satıra bakacak olursak K harfinin şifresiz metnin 11. harfi olduğunu görürüz. Dosyaya yazdırırken ilk indis 1 olarak kabul edilmiştir.

Bu şifreyi çözmek için harfleri doğru bir şekilde sıralarsak BURSATEKNİKUNIVERSİTESİ metnini elde ederiz.

Ali, gerekli kodun çok kısa bir kısmını yazabildi. Aşağıdaki şıklara vereceğiniz cevaplarla ona yardımcı olunuz.

```
void main(){
    int x, sayac = 0;
    char y, *karakterler;
    FILE *fptr;
```

11	K
23	I
8	K
5	A
4	S
17	R
10	I
14	I
7	E
2	U
1	B
19	I
22	S
20	T
21	E
3	R
9	N
15	V
18	S
13	N
12	U
6	T
16	E

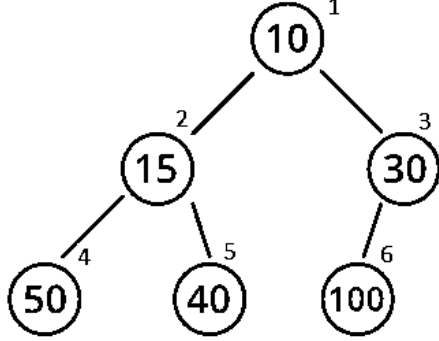
a) [8p] “sifreli.txt” dosyasını okuma modunda açınız. Dosya içerisindeki satır sayısını bir döngü yardımıyla sayarak *sayac* isimli değişkene kaydediniz. Ardından, dosyayı kapatmadan dosyanın başına dönünüz.

```
fptr = fopen("sifreli.txt", "r");
while(!feof(fptr)) {
    fscanf(fptr, "%d\t%c\n", &x, &y);
    sayac++;
}
rewind(fptr);
```

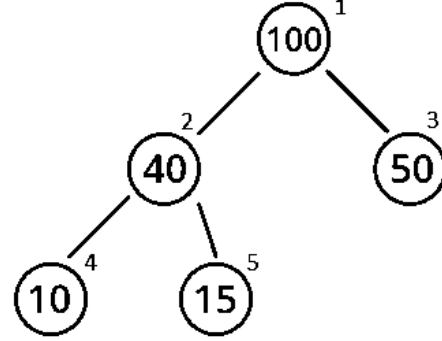
b) [12p] *sayac+1* uzunluklu *karakterler* dizisini dinamik olarak yaratınız. Ardından, yine bir döngü yardımıyla dosyanın her bir satırındaki indis ve harf çiftini okuyup okuduğunuz harfi, indis değerine uygun bir şekilde *karakterler* dizisine yerleştiriniz. Son olarak da *karakterler* dizisini ekrana yazdırıp dosyayı kapatınız.

```
karakterler = (char *) malloc(sizeof(char) * (1 + sayac));
while(!feof(fptr)){
    fscanf(fptr, "%d\t%c\n", &x, &y);
    karakterler[x - 1] = y;
}
karakterler[sayac] = '\0';
puts(karakterler);
fclose(fptr);
```

5) [25p] Aşağıdaki şekilde “Heap Organizasyonu” adı verilen bir yapı görülmektedir. Bu yapıda daire ile gösterilen düğümler ve her düğümün içerisinde de bazı rasgele sayılar görülmektedir. Her düğümün sağ üst köşesinde o düğümün indisini gösteren bir sayı vardır. Her ata düğümün, kendisinden türeyen en fazla 2 adet çocuk düğümü vardır. Örneğin, soldaki Min Heap organizasyonu için 2 ve 3 numaralı düğümler 1 numaralı ata düğümün çocuklarıdır.



Min Heap



Max Heap

Min Heap organizasyonunda çocuk düğümlerin içinde yazan sayılar ata düğümde yazan sayıdan büyük olmak zorundadır. Max Heap organizasyonunda ise tam tersi bir durum söz konusudur. Aynı atanın çocukları olan kardeş düğümlerin birbirlerine göre büyüklüğüne küçüklüğüne bakılmaz.

Her Heap organizasyonu, yukarıdan aşağıya ve soldan sağa doğru yapılan bir tarama sonucunda bir sayı dizisi ile ifade edilebilir. Örneğin, Min Heap örneği yukarıdan aşağıya ve soldan sağa doğru eleman eleman tarandığında {10, 15, 30, 50, 40, 100} dizisi elde edilir. Max Heap örneği için ise dizimiz {100, 40, 50, 10, 15} şeklinde olur.

Her ata düğümün i gibi bir indisi varsa çocuklarının $2i$ ve $2i + 1$ şeklinde indisleri olur.

Verilen bilgiler ışığında, kullanıcıdan aldığı bir sayı dizisinin Min Heap ya da Max Heap olduğuna, veya ikisi de olmadığına karar veren bir C programı yazınız. Programınızın örnek ekran çıktısı aşağıdakine benzemelidir:

1. Deneme	2. Deneme	3. Deneme
Heapin eleman sayısını giriniz: 6 1. elemanı giriniz: 10 2. elemanı giriniz: 15 3. elemanı giriniz: 30 4. elemanı giriniz: 50 5. elemanı giriniz: 40 6. elemanı giriniz: 100 Bu dizi bir min heap!	Heapin eleman sayısını giriniz: 5 1. elemanı giriniz: 100 2. elemanı giriniz: 40 3. elemanı giriniz: 50 4. elemanı giriniz: 10 5. elemanı giriniz: 15 Bu dizi bir max heap!	Heapin eleman sayısını giriniz: 5 1. elemanı giriniz: 100 2. elemanı giriniz: 40 3. elemanı giriniz: 101 4. elemanı giriniz: 10 5. elemanı giriniz: 15 Bu dizi ne min heap ne de max heap!

İpucu: int minheapmi(...) ve int maxheapmi(...) gibi birbirine çok benzeyen iki fonksiyon yazarak test işlemi daha kolay yapabilirsiniz!

Soru 5 yanıt:

```
int minheapmi(int dizi[],int n){
    int minheap=1;
    for(int i=1;i<=n;i++){
        if((2*i+1)<=n){ //iki çocuğu varsa
            if(dizi[i-1]>=dizi[2*i-1]||dizi[i-1]>=dizi[2*i]){
                minheap=0;
                break;
            }
        }
        else if((2*i)<=n){ //tek çocuğu varsa
            if(dizi[i-1]>=dizi[2*i-1]){
                minheap=0;
                break;
            }
        }
    }
    return minheap;
}
```

```
int maxheapmi(int dizi[],int n){ //minheapmi()'nin kopyası
    int maxheap=1;
    for(int i=1;i<=n;i++){
        if((2*i+1)<=n){ //iki çocuğu varsa
            if(dizi[i-1]<=dizi[2*i-1]||dizi[i-1]<=dizi[2*i]){
                maxheap=0;
                break;
            }
        }
        else if((2*i)<=n){ //tek çocuğu varsa
            if(dizi[i-1]<=dizi[2*i-1]){
                maxheap=0;
                break;
            }
        }
    }
    return maxheap;
}
```

```
int main(){
    int n, *dizi;
    printf("Heapin eleman sayisini giriniz: ");
    scanf("%d",&n);

    printf("\n");

    dizi=(int*)malloc(n*sizeof(int));

    for(int i=1;i<=n;i++){
        printf("%d. elemani giriniz: ",i);
        scanf("%d",&dizi[i-1]);
    }

    printf("\n");
}
```

```
    if(minheapmi(dizi,n))
        printf("Bu dizi bir min heap!");
    else if(maxheapmi(dizi,n))
        printf("Bu dizi bir max heap!");
    else
        printf("Bu dizi ne min heap ne de max heap!");

    return 0;
}
```