

Ödev raporlarının hazırlanması ve sisteme yüklenmesi hakkında uyulacak kurallar!

Ödevler Word dokümanı olarak aşağıdaki kurallara uygun biçimde hazırlanacak, sisteme yüklenmeden önce pdf formatına çevrilecektir.

Aşağıdaki tablo araçlarıyla oluşturulan şablon kullanılacaktır. Sayfa kenar boşlukları 1,27cm (dar) biçimde seçilmelidir, yazı font boyutu 11'den büyük olmamalıdır, ödevler sıralı olmalıdır. Ödevler OBS sınav modülü üzerinden tanımlanacak zaman aralığında sisteme yüklenmelidir. Daha sonra email ile veya başka bir kanaldan gönderilmemelidir. Sisteme .exe dosyalar yüklenmemelidir. .cpp kaynak dosyaları yüklenmelidir. Sisteme .pdf ve .cpp dosyaları bir klasör içerisinde birlikte bulunmalı, sisteme yüklenmeden önce zip veya rar programları ile sıkıştırılarak tek dosya haline dönüştürülmelidir. Aşağıdaki tablo şablonunu boş bir sayfaya yapıştırarak, gerektiğinde tabloya yeni satırlar ekleyerek düzgün, standart bir ödev raporu oluşturmak için itina gösteriniz. Ödevlerin Finale olan katkı yüzdesi dönem sonundaki toplam ödev sayısına göre belirlenecektir.

Ödevlerin programlama eğitiminize katkısı olabilmesi için öneriler;

1.aşama: Ödevleri kendinize tanıdığınız makul bir süre içerisinde kendiniz yalnız yoğunlaşarak, önce kağıt üzerinde analiz, tasarım, kaba kod, kod çalışması yapınız. Problemi yapısına uygun, mantıksal sıralı, küçük birkaç parçaya bölerek çözümleyin, kodlayın, çalıştırarak ilerleyin. Yazdığınız program tamamlanana kadar unutmayın sürekli test aşamasındadır!

2.aşama: Makul süre sonunda daha kısa bir süre ayırarak internet kaynaklarından takıldığınız bölümlerin çözümlerini araştırınız. Unutmayın internette aynı ödevin çözümü yoktur! İnternetteki kaynaklardan, örneklerden teknik bilgi araştırınız.

3.aşama: Sınıf arkadaşlarınızla bilgi paylaşımına gidiniz. Bu paylaşım kesinlikle kopyalama, dosya değişimi seviyesine çıkması! Bu tür paylaşımlar size ve arkadaşına katkı sağlamaz!

4.aşama: Hiç veya %60'a kadar katkı olmayan çözümü ödev olarak yükleme! Boş kalsın, kendini kandırma! Gönderdiğiniz ödevleri sizin yaptığınızı ve konuyu öğrendiğinizi farz ediyoruz.

Başarılar diliyorum. Prof.Dr.Tuncay AYDOĞAN

ÖDEV 1: Bir bağlı dairesel listeler için **cutlast()** adında bir fonksiyonunu yazınız. Ana programda deneyiniz. Bu fonksiyonu standart fonksiyonları kullanmadan da yazabilirsiniz.

ÖDEV 2: data bölümünde {string ad; string soyad; string no; short yas;} bulunan bir bağlı doğrusal liste için öğrenci adında bir node tanımlayınız. Kendisine parametre olarak gelen sınıf adındaki bir bağlı doğrusal listeden yine kendisine parametre olarak gelen yaştaki öğrenci sayısını hesaplayan ve bu öğrencilerin bilgilerini ekrana yazan **ogrencisorgula()** adında bir fonksiyonunu yazınız. Ana programda deneyiniz.

ÖDEV 3: Kendisine parametre olarak gelen bir bağlı dairesel listeyi bir bağlı doğrusal listeye dönüştüren ve çağrıldığı yere geriye gönderen **donustur()** adında bir fonksiyon yazınız. Ana programda deneyiniz. (Geçen senenin sınav sorusu)

ÖDEV 4: Kendisine parametre olarak gelen iki adet bir bağlı doğrusal liste birbirinin kopyası ise geriye true, değilse geriye false gönderen **karsilastir()** adında bir fonksiyon yazınız. Ana programda deneyiniz. (Geçen senenin sınav sorusu)

ÖDEV 5: Kendisine parametre olarak gelen bir bağlı dairesel listedeki nodelerin datalarının ortalamasını hesaplayıp geriye gönderen **ortalamaal()** adında bir fonksiyonunu yazınız. Ana programda deneyiniz.

ÖDEV 6: data bölümünde {integer sayfa_sayisi; string kitap_adi;} bulunan bir bağlı doğrusal liste için "kitap" adında bir veri yapısı tanımlayınız. Kendisine parametre olarak gelen kitap veri yapısından tanımlı "kutuphane" adındaki bir bağlı doğrusal listedeki en çok sayfa sayısına sahip kitabı çağrıldığı yere geri gönderen **kalinkitap()** adında bir fonksiyon yazınız. Ana programda bu fonksiyonu deneyiniz ve sonucu ekrana yazdırınız.

ÖDEV 7: 6.ödevdeki sorudaki fonksiyonu recursive fonksiyon olarak programlayınız.

ISPARTA UYGULAMALI BİLİMLER ÜNİVERSİTESİ -TEKNOLOJİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
2024-2025 BAHAR DÖNEMİ BLG-102 VERİ YAPILARI DERSİ ÖDEV RAPORU

Ad Soyad: CEREN MITIRIK
Numara: 2212721032

(Bu bölüme ödev numarasını ve ödevi yazın.)

ÖDEV : 1.SORUNUN KODLARI:

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

// Listeye eleman ekleyen fonksiyon (sona ekleme)
void ekle(Node*& head, int veri) {
    Node* yeni = new Node;
    yeni->data = veri;
    yeni->next = NULL;

    if (head == NULL) {
        head = yeni;
        head->next = head;
    } else {
        Node* temp = head;
        while (temp->next != head) {
            temp = temp->next;
        }
        temp->next = yeni;
        yeni->next = head;
    }
}

// Listenin son düğümünü silen fonksiyon
void cutlast(Node*& head) {
    if (head == NULL) {
        cout << "Liste bos.\n";
        return;
    }

    // Tek eleman varsa
    if (head->next == head) {
        delete head;
        head = NULL;
        return;
    }

    Node* onceki = NULL;
    Node* temp = head;

    while (temp->next != head) {
        onceki = temp;
        temp = temp->next;
    }

    onceki->next = head;
    delete temp;
}
```

```

}

// Listeyi yazdıran fonksiyon
void yazdir(Node* head) {
    if (head == NULL) {
        cout << "Liste bos.\n";
        return;
    }

    Node* temp = head;
    do {
        cout << temp->data << " -> ";
        temp = temp->next;
    } while (temp != head);
    cout << "(basa doner)\n";
}

// Ana program
int main() {
    Node* liste = NULL;

    // Eleman ekleyelim
    ekle(liste, 10);
    ekle(liste, 20);
    ekle(liste, 30);
    ekle(liste, 40);

    cout << "Ilk hali:\n";
    yazdir(liste);

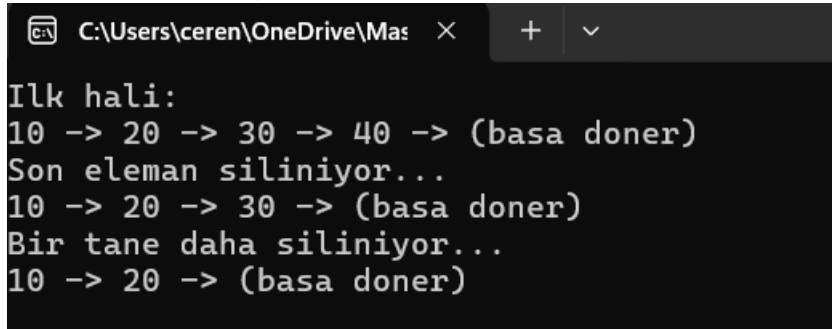
    cout << "Son eleman siliniyor...\n";
    cutlast(liste);
    yazdir(liste);

    cout << "Bir tane daha siliniyor...\n";
    cutlast(liste);
    yazdir(liste);

    return 0;
}

```

1.SORUNUN EKRAN ÇIKTISI:



```

C:\Users\ceren\OneDrive\Mas
Ilk hali:
10 -> 20 -> 30 -> 40 -> (basa doner)
Son eleman siliniyor...
10 -> 20 -> 30 -> (basa doner)
Bir tane daha siliniyor...
10 -> 20 -> (basa doner)

```

(Bu bölüme sıradaki ödev numarasını ve ödevi yazın.)

ÖDEV : 2.SORUNUN KODLARI: #include <iostream>
using namespace std;

// Öğrenci düğümü yapısı

```

struct ogrenci {
    string ad;
    string soyad;
    string no;
    short yas;
    ogrenci* next;
};

// Listeye yeni öğrenci ekler (listenin sonuna)
ogrenci* ogrenciEkle(ogrenci* bas, string ad, string soyad, string no, short yas) {
    ogrenci* yeni = new ogrenci;
    yeni->ad = ad;
    yeni->soyad = soyad;
    yeni->no = no;
    yeni->yas = yas;
    yeni->next = NULL;

    if (bas == NULL) {
        return yeni;
    } else {
        ogrenci* temp = bas;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = yeni;
        return bas;
    }
}

// Verilen yaştaki öğrencileri ekrana yazdırır ve sayısını gösterir
void ogrencisorgula(ogrenci* bas, short arananYas) {
    int sayac = 0;
    ogrenci* temp = bas;

    while (temp != NULL) {
        if (temp->yas == arananYas) {
            cout << "Ad: " << temp->ad
                << ", Soyad: " << temp->soyad
                << ", No: " << temp->no
                << ", Yas: " << temp->yas << endl;
            sayac++;
        }
        temp = temp->next;
    }

    cout << "\n" << arananYas << " yasındaki ogrenci sayısı: " << sayac << endl;
}

// Ana fonksiyon
int main() {
    ogrenci* sinif = NULL;

    sinif = ogrenciEkle(sinif, "Ali", "Yılmaz", "101", 20);
    sinif = ogrenciEkle(sinif, "Ayşe", "Demir", "102", 21);
    sinif = ogrenciEkle(sinif, "Mehmet", "Kaya", "103", 20);

    short yas;

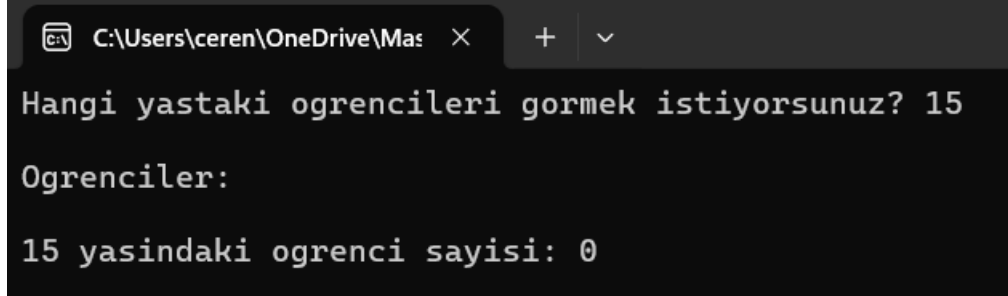
```

```
cout << "Hangi yastaki ogrencileri gormek istiyorsunuz? ";
cin >> yas;

cout << "\nOgrenciler:\n";
ogrencisorgula(sinif, yas);

return 0;
}
```

2.SORUNUN EKRAN ÇIKTISI:



```
C:\Users\ceren\OneDrive\Mas × + ∨

Hangi yastaki ogrencileri gormek istiyorsunuz? 15

Ogrenciler:

15 yasindaki ogrenci sayisi: 0
```

ÖDEV: 3.SORUNUN KODLARI:

```
#include <iostream>
using namespace std;

// Düğüm yapısı
struct Node {
    int data;
    Node* next;
};

// Dairesel bağlı listeyi doğrusal bağlı listeye dönüştüren fonksiyon
Node* donustur(Node* bas) {
    if (bas == NULL)
        return NULL;

    Node* temp = bas;

    while (temp->next != bas) {
        temp = temp->next;
    }

    temp->next = NULL; // Daireselliği sona erdir

    return bas; // Artık doğrusal listenin başı
}

// Listeyi yazdırma fonksiyonu
void yazdir(Node* bas) {
    Node* temp = bas;
    while (temp != NULL) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

// Ana program
```

```

int main() {
    // Dairesel bağlı liste oluştur
    Node* n1 = new Node{10, NULL};
    Node* n2 = new Node{20, NULL};
    Node* n3 = new Node{30, NULL};

    n1->next = n2;
    n2->next = n3;
    n3->next = n1; // Daireselliği tamamla

    cout << "Dönüştürmeden önce dairesel liste (ilk 3 eleman):\n";
    Node* temp = n1;
    for (int i = 0; i < 3; i++) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "(basa döner)\n";

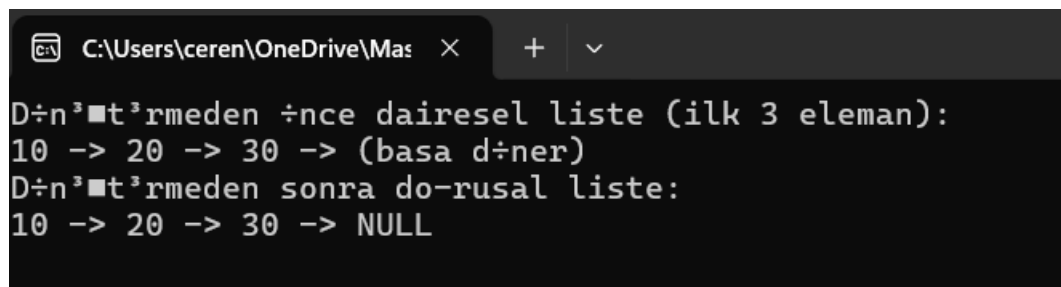
    // Dönüştür
    Node* dogrusal = donustur(n1);

    cout << "Dönüştürmeden sonra doğrusal liste:\n";
    yazdir(dogrusal);

    return 0;
}

```

3.SORUNUN EKRAN ÇIKTISI:



```

C:\Users\ceren\OneDrive\Mas
Dönüştürmeden önce dairesel liste (ilk 3 eleman):
10 -> 20 -> 30 -> (basa döner)
Dönüştürmeden sonra doğrusal liste:
10 -> 20 -> 30 -> NULL

```

ÖDEV:4.SORUNUN KODLARI:

```

#include <iostream>
using namespace std;

// Düğüm yapısı
struct Node {
    int data;
    Node* next;
};

// İki bağlı listeyi karşılaştıran fonksiyon
bool karsilastir(Node* liste1, Node* liste2) {
    while (liste1 != NULL && liste2 != NULL) {
        if (liste1->data != liste2->data) {
            return false; // Veri uyuşmuyorsa aynı değildir
        }
    }
}

```

```

    liste1 = liste1->next;
    liste2 = liste2->next;
}

// Her ikisi de aynı anda NULL olmalı (uzunluk da aynı olmalı)
return (liste1 == NULL && liste2 == NULL);
}

// Eleman ekleme (listenin sonuna)
void sonaEkle(Node*& bas, int veri) {
    Node* yeni = new Node{veri, NULL};
    if (bas == NULL) {
        bas = yeni;
    } else {
        Node* temp = bas;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = yeni;
    }
}

// Listeyi yazdıran yardımcı fonksiyon
void yazdir(Node* bas) {
    Node* temp = bas;
    while (temp != NULL) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

// Ana program
int main() {
    Node* liste1 = NULL;
    Node* liste2 = NULL;

    // Her iki listeye aynı elemanları ekleyelim
    sonaEkle(liste1, 10);
    sonaEkle(liste1, 20);
    sonaEkle(liste1, 30);

    sonaEkle(liste2, 10);
    sonaEkle(liste2, 20);
    sonaEkle(liste2, 30);

    cout << "Liste 1: ";
    yazdir(liste1);

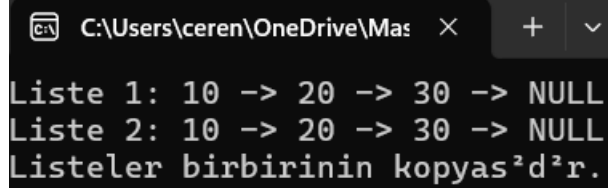
    cout << "Liste 2: ";
    yazdir(liste2);

    // Karşılaştırma
    if (karsilastir(liste1, liste2)) {
        cout << "Listeler birbirinin kopyasıdır.\n";
    } else {
        cout << "Listeler birbirinden farklıdır.\n";
    }
}

```

```
}  
  
return 0;  
}
```

4.SORUNUN EKRAN ÇIKTISI:



```
Liste 1: 10 -> 20 -> 30 -> NULL  
Liste 2: 10 -> 20 -> 30 -> NULL  
Listeler birbirinin kopyas^d^r.
```

ÖDEV:5.SORUNUN KODLARI:

```
#include <iostream>  
using namespace std;  
  
// Bağlı liste düğüm yapısı  
struct Node {  
    int data;  
    Node* next;  
};  
  
// Ortalama alma fonksiyonu  
float ortalamaal(Node* head) {  
    if (head == NULL) return 0; // Eğer liste boşsa, 0 döndür  
    int toplam = 0, sayac = 0;  
    Node* current = head;  
  
    // Bağlı dairesel listenin tamamını dolaş  
    do {  
        toplam += current->data; // Geçerli düğümün verisini topla  
        sayac++; // Sayacı artır  
        current = current->next; // Bir sonraki düğüme geç  
    } while (current != head); // Listenin başına dönene kadar devam et  
  
    return static_cast<float>(toplam) / sayac; // Ortalamayı hesapla  
}  
  
// Ana program  
int main() {  
    // Bağlı dairesel listeyi oluşturma  
    Node* head = new Node;  
    head->data = 10;  
  
    Node* second = new Node;  
    second->data = 20;  
    head->next = second;  
  
    Node* third = new Node;  
    third->data = 30;  
    second->next = third;  
  
    // Dairesel bağlantı yapıyoruz
```



```

third->next = head; // Son düğüm başa bağlanıyor

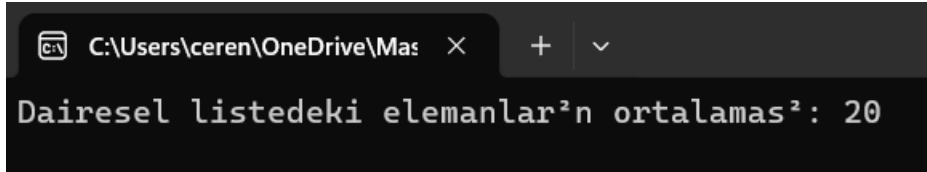
// Ortalamayı hesapla
float ortalama = ortalamaal(head);

// Sonucu ekrana yazdır
cout << "Dairesel listedeki elemanların ortalaması: " << ortalama << endl;

return 0;
}

```

5.SORUNUN EKRAN ÇIKTISI:



```

C:\Users\ceren\OneDrive\Mas >
Dairesel listedeki elemanlar²n ortalamas²: 20

```

ÖDEV: 6.SORUNUN KODLARI:

```

#include <iostream>
using namespace std;

// Kitap yapısı
struct kitap {
    int sayfa_sayisi;
    string kitap_adi;
    kitap* next; // Bir sonraki kitaba işaretçi
};

// En çok sayfa sayısına sahip kitabı bulan fonksiyon
kitap* kalinkitap(kitap* kutuphane) {
    if (kutuphane == NULL) {
        return NULL; // Eğer liste boşsa NULL döndür
    }

    kitap* enKalin = kutuphane; // Başlangıç olarak ilk kitabı al
    kutuphane = kutuphane->next; // Sonraki kitaplara geç

    while (kutuphane != NULL) {
        if (kutuphane->sayfa_sayisi > enKalin->sayfa_sayisi) {
            enKalin = kutuphane; // Daha fazla sayfa varsa güncelle
        }
        kutuphane = kutuphane->next; // Bir sonraki kitaba geç
    }

    return enKalin; // En fazla sayfa sayısına sahip kitabı geri döndür
}

// Ana program
int main() {
    // 3 kitaplık bağlı liste oluşturalım
    kitap* k1 = new kitap{250, "Kitap A", NULL};
    kitap* k2 = new kitap{320, "Kitap B", NULL};
    kitap* k3 = new kitap{180, "Kitap C", NULL};

```

```

// Liste bağlantılarını kur
k1->next = k2;
k2->next = k3;
k3->next = NULL;

// En kalın kitabı bul
kitap* enKalin = kalinkitap(k1);

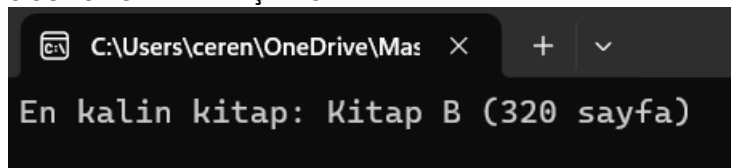
// Sonucu ekrana yazdır
if (enKalin != NULL) {
    cout << "En kalın kitap: " << enKalin->kitap_adi
        << " (" << enKalin->sayfa_sayisi << " sayfa)" << endl;
} else {
    cout << "Kutuphane bos." << endl;
}

// Belleği temizleyelim
delete k1;
delete k2;
delete k3;

return 0;
}

```

6.SORUNUN EKRAN ÇIKTISI:



```

C:\Users\ceren\OneDrive\Mas x + v
En kalın kitap: Kitap B (320 sayfa)

```

ÖDEV: 7.SORUNUN KODLARI:

```

#include <iostream>
using namespace std;

// Kitap yapısı
struct kitap {
    int sayfa_sayisi;
    string kitap_adi;
    kitap* next;
};

// Recursive olarak en çok sayfalı kitabı bulan fonksiyon
kitap* kalinkitap(kitap* kutuphane) {
    // Eğer listede sadece bir eleman varsa veya son elemana geldiysek
    if (kutuphane == NULL || kutuphane->next == NULL)
        return kutuphane;

    // Listeyi geriye doğru gez ve diğer kitaplarla karşılaştıır
    kitap* enKalan = kalinkitap(kutuphane->next);

    if (kutuphane->sayfa_sayisi > enKalan->sayfa_sayisi)
        return kutuphane;
    else
        return enKalan;
}

// Ana program
int main() {

```

```
// 3 kitaplık bağılı liste oluşturalım
kitap* k1 = new kitap{250, "Kitap A", NULL};
kitap* k2 = new kitap{320, "Kitap B", NULL};
kitap* k3 = new kitap{180, "Kitap C", NULL};

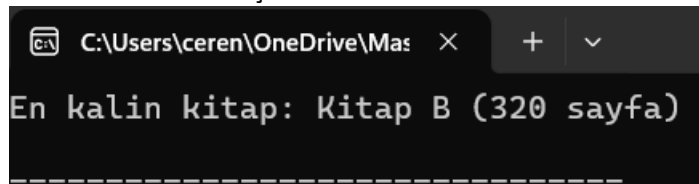
// Liste bağlantılarını kur
k1->next = k2;
k2->next = k3;
k3->next = NULL;

// En kalın kitabı bul
kitap* enKalin = kalinkitap(k1);

// Sonucu ekrana yazdır
if (enKalin != NULL) {
    cout << "En kalın kitap: " << enKalin->kitap_adi
        << " (" << enKalin->sayfa_sayisi << " sayfa)" << endl;
} else {
    cout << "Kutuphane bos." << endl;
}

return 0;
}
```

7.SORUNUN EKRAN ÇIKTISI:



The screenshot shows a Windows command prompt window with a dark background. The title bar at the top indicates the file path 'C:\Users\ceren\OneDrive\Mas' and includes standard window controls (close, maximize, minimize). The main content area displays the output of the program: 'En kalın kitap: Kitap B (320 sayfa)'. Below the output, there is a dashed line indicating the end of the output.