

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 çıkmasın! Bu tür paylaşımlar size ve arkadaşına katkı sağlamaz!

4.aşama: Hiç veya %60'a kadar katkın 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 listelere ait ders notlarını içeren fonksiyonları bbdal.h adında bir headerfile olarak programlayınız. Tüm fonksiyonları bir main() fonksiyonda test ediniz (deneyiniz). Fonksiyonların deneme sonuçlarını raporda kısa açıklamalı veriniz.

ÖDEV 2: 1.ödevde oluşturduğunuz bbdal.h adındaki headerfile kullanarak, kendisine parametre olarak gelen değeri NULL olan bir bağlı doğrusal listeye 10 adet ve 50-500 arasında rastgele dataları olan node ekleyerek bir liste oluşturan createlist() adında bir fonksiyon yazınız , ana programda deneyiniz. Deneme sonuçlarını dumplist() ile test ediniz.

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;
```

```
// Düğüm yapısı
```

```
struct Node {
```

```
    int data;
```

```
    Node* next;
```

```
};
```

```
// Bağlı Dairesel Listeyi başlatmak için fonksiyon
```

```
Node* createList() {
```

```
    return NULL; // Liste başlangıçta boş
```

```
}
```

```
// Listeye eleman eklemek için fonksiyon
```

```
void append(Node*& head, int data) {
```

```
    Node* newNode = new Node;
```

```
    newNode->data = data;
```

```
    newNode->next = NULL;
```

```
    if (head == NULL) {
```

```
        head = newNode;
```

```
        head->next = head; // İlk elemanın kendisine işaret etmesi sağlanır
```

```
    } else {
```

```
        Node* temp = head;
```

```
        while (temp->next != head) {
```

```
            temp = temp->next;
```

```
        }
```

```
        temp->next = newNode;
```

```
        newNode->next = head; // Yeni eleman başa bağlanacak şekilde sonlanır
```

```
    }
```

```
}
```

```
// Son düğümü silme fonksiyonu (cutlast)
```

```
void cutlast(Node*& head) {
```

```
    if (head == NULL) {
```

```
        cout << "Liste boş!" << endl;
```

```
        return;
```

```
    }
```

```
// Tek eleman varsa
```

```
if (head->next == head) {
```

```
    delete head;
```

```
    head = NULL;
```

```
    return;
```

```
}
```

```
Node* temp = head;
```

```
Node* prev = NULL;
```

```
while (temp->next != head) {
```

```
    prev = temp;
```

```
    temp = temp->next;
}

prev->next = head; // Sondan bir önceki eleman başa bağlanır
delete temp; // Son eleman silinir
}
```

// Listeyi yazdırma fonksiyonu

```
void display(Node* head) {
    if (head == NULL) {
        cout << "Liste boş!" << endl;
        return;
    }

    Node* temp = head;
    do {
        cout << temp->data << " -> ";
        temp = temp->next;
    } while (temp != head);
    cout << "(tekrar başa döner)" << endl;
}
```

// Belleği temizleme fonksiyonu (isteğe bağlı)

```
void deleteList(Node*& head) {
    while (head != NULL) {
        cutlast(head);
    }
}
```

// Ana program

```
int main() {
    Node* list = createList(); // Bağlı dairesel listeyi oluştur

    cout << "Listeye 10, 20, 30 ve 40 ekliyoruz..." << endl;
    append(list, 10);
    append(list, 20);
    append(list, 30);
    append(list, 40);

    cout << "Listeyi yazdıralım:" << endl;
    display(list); // Listeyi yazdır

    cout << "Son elemanı siliyoruz..." << endl;
    cutlast(list); // Son elemanı sil

    cout << "Listeyi yazdıralım:" << endl;
    display(list); // Yeni listeyi yazdır

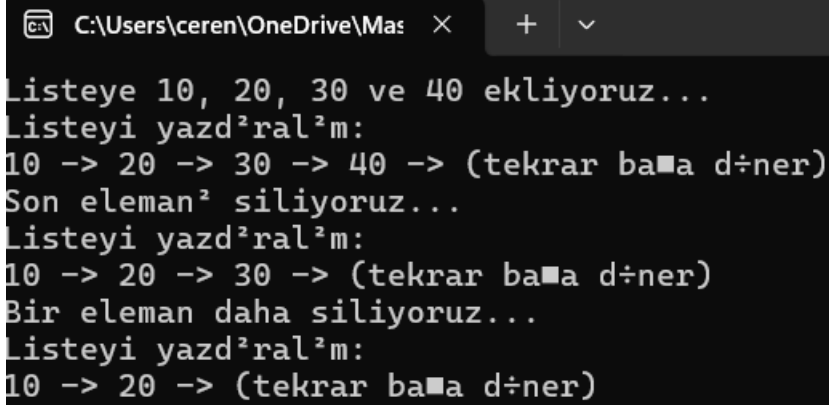
    cout << "Bir eleman daha siliyoruz..." << endl;
    cutlast(list); // Tekrar bir eleman sil

    cout << "Listeyi yazdıralım:" << endl;
    display(list); // Yeni listeyi yazdır

    // Bellek temizleme
    deleteList(list);
}
```

```
return 0;
}
```

1.SORUNUN EKRAN ÇIKTISI:



```
C:\Users\ceren\OneDrive\Mas x + v
Listeye 10, 20, 30 ve 40 ekliyoruz...
Listeyi yazdıralım:
10 -> 20 -> 30 -> 40 -> (tekrar başla dener)
Son elemanı siliyoruz...
Listeyi yazdıralım:
10 -> 20 -> 30 -> (tekrar başla dener)
Bir eleman daha siliyoruz...
Listeyi yazdıralım:
10 -> 20 -> (tekrar başla dener)
```

ÖDEV:2.SORUNUN KODLARI:

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

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

// Bağlı listeyi başlatmak için fonksiyon
Node* createList() {
    return NULL; // Liste başlangıçta boş
}

// Listeye eleman eklemek için fonksiyon
void append(Node*& head, int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = NULL;

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

// Listeyi yazdırma fonksiyonu
void dumpList(Node* head) {
    if (head == NULL) {
        cout << "Liste boş!" << endl;
    }
}
```

```

    return;
}

Node* temp = head;
while (temp != NULL) {
    cout << temp->data << " -> ";
    temp = temp->next;
}
cout << "NULL" << endl;
}

// Rastgele baęlı liste oluřturma fonksiyonu
void createRandomList(Node*& head) {
    srand(time(0)); // Rastgele sayı üretici için seed

    for (int i = 0; i < 10; i++) {
        int randomData = rand() % 451 + 50; // 50 ile 500 arasında rastgele sayı
        append(head, randomData);
    }
}

// Ana program
int main() {
    Node* list = createList(); // Boř bir baęlı liste oluřtur

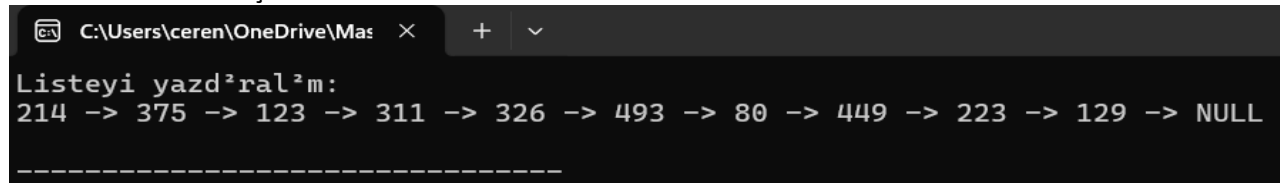
    // Rastgele verilerle listeyi oluřtur
    createRandomList(list);

    // Listeyi yazdır
    cout << "Listeyi yazdırılım:" << endl;
    dumpList(list);

    return 0;
}

```

2.SORUNUN EKRAN ÇIKTISI:



```

C:\Users\ceren\OneDrive\Mas
Listeyi yazdırılım:
214 -> 375 -> 123 -> 311 -> 326 -> 493 -> 80 -> 449 -> 223 -> 129 -> NULL
-----

```