

ISPARTA UYGULAMALI BİLİMLER ÜNİVERSİTESİ - TEKNOLOJİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ
BÖLÜMÜ

2024-2025 BAHAR DÖNEMİ BLG-102 VERİ YAPILARI DERSİ ÖDEV RAPORU

Ad Soyad: Ceren Mıtırık

Numara: 2212721032

(Bu bölüme ödev numarasını ve ödevi yazın.)

ÖDEV :

ÖDEV -1 :

TEST-1:

```
Bos liste yazdirma:  
Liste bos!
```

Liste başlangıçta boş gösterilir. dumplist() fonksiyonunun boş listeleri nasıl gösterdiği verilir.

TEST-2:

```
Yeni dugum olusturuldu: 100
```

newnode() ile hafızada yeni bir düğüm ayır. Düğümün data alanına 100 değeri ata.

TEST-3:

```
Basa 100 eklendi:  
1.Dugum - Adres: 0xbb1540, Veri: 100, Sonraki: 0
```

addhead() ile tek düğümlü liste oluştur. Listenin başına eklenen düğümün link kısmının NULL olduğunu gör.

TEST-4:

```
Son dugumun verisi: 100
```

Tek elemanlı listede last() fonksiyonunun ilk düğümü döndürmesini test et. Fonksiyonun null kontrolunu yap.

TEST -5:

```
10,20,30 eklendi:  
1.Dugum - Adres: 0xbb1a40, Veri: 30, Sonraki: 0xbb1580  
2.Dugum - Adres: 0xbb1580, Veri: 20, Sonraki: 0xbb1560  
3.Dugum - Adres: 0xbb1560, Veri: 10, Sonraki: 0xbb1540  
4.Dugum - Adres: 0xbb1540, Veri: 100, Sonraki: 0
```

cons() fonksiyonuyla 3 düğüm oluştur. addhead() ile ters sırada eklendiğini listele.

TEST- 6 :

```
20 degeri bulundu
```

locate() ile 20 değerinin listede bulunup bulunmadığı kontrol et. Bulunan düğümün adresini yazdır.

TEST -7:

```
node3 listede var
```

member() fonksiyonuyla belirli bir düğüm adresinin listede olup olmadığı test et.

TEST- 8:

```
Bastan kesilen: 30
Yeni liste:
1.Dugum - Adres: 0xbb1580, Veri: 20, Sonraki: 0xbb1560
2.Dugum - Adres: 0xbb1560, Veri: 10, Sonraki: 0xbb1540
3.Dugum - Adres: 0xbb1540, Veri: 100, Sonraki: 0
```

cuthead() ile listenin başındaki 30 değerli düğümünü kes. Kesilen düğümün linkinin NULL yapıldığı ve listenin 20 ile başladığını gör.

TEST- 9:

```
Kopya liste:
1.Dugum - Adres: 0xbb1a60, Veri: 30, Sonraki: 0xbb1a80
2.Dugum - Adres: 0xbb1a80, Veri: 20, Sonraki: 0xbb1aa0
3.Dugum - Adres: 0xbb1aa0, Veri: 10, Sonraki: 0xbb1ac0
4.Dugum - Adres: 0xbb1ac0, Veri: 100, Sonraki: 0
```

copy() fonksiyonuyla listenin bir kopyası oluştur. Kopya listeyi göster.

TEST- 10 :

```
İlerletme sonrasi veri: 20
```

advance() ile listenin ikinci düğümüne geç. Fonksiyonun listenin sonuna gelince false döndüğü göster.

TEST-11 :

```
Birlestirme sonrasi:
1.Dugum - Adres: 0xbb1a40, Veri: 30, Sonraki: 0xbb1580
2.Dugum - Adres: 0xbb1580, Veri: 20, Sonraki: 0xbb1560
3.Dugum - Adres: 0xbb1560, Veri: 10, Sonraki: 0xbb1540
4.Dugum - Adres: 0xbb1540, Veri: 100, Sonraki: 0xbb1ae0
5.Dugum - Adres: 0xbb1ae0, Veri: 40, Sonraki: 0xbb1b00
6.Dugum - Adres: 0xbb1b00, Veri: 50, Sonraki: 0
```

40 ve 50 değerlerinden oluşan yeni bir liste concatenate() ile ana listeye ekle. Doğru eklenmiş mi kontrol et.

TEST-12:

```
silindi:
1.Dugum - Adres: 0xbb1a40, Veri: 30, Sonraki: 0xbb1560
2.Dugum - Adres: 0xbb1560, Veri: 10, Sonraki: 0xbb1540
3.Dugum - Adres: 0xbb1540, Veri: 100, Sonraki: 0xbb1ae0
4.Dugum - Adres: 0xbb1ae0, Veri: 40, Sonraki: 0xbb1b00
5.Dugum - Adres: 0xbb1b00, Veri: 50, Sonraki: 0
```

deletenode() ile 20 değerli düğüm adresle bulunup sil. Diğer verilerin bağlantısını düzenle.

TEST-13 :

```
Tek dugum silme oncesi: 99
Silme sonrasi pointer: 0
```

Tek düğümü sil. Silinen pointerin Null yaptığını göster.

ÖDEV -2:

Oluşturulan Liste:

```
1. Dugum - Veri: 183, Adres: 0x761b00, Sonraki: 0x761ae0
2. Dugum - Veri: 282, Adres: 0x761ae0, Sonraki: 0x761ac0
3. Dugum - Veri: 206, Adres: 0x761ac0, Sonraki: 0x761aa0
4. Dugum - Veri: 166, Adres: 0x761aa0, Sonraki: 0x761a80
5. Dugum - Veri: 274, Adres: 0x761a80, Sonraki: 0x761a60
6. Dugum - Veri: 109, Adres: 0x761a60, Sonraki: 0x761a40
7. Dugum - Veri: 328, Adres: 0x761a40, Sonraki: 0x761580
8. Dugum - Veri: 137, Adres: 0x761580, Sonraki: 0x761560
9. Dugum - Veri: 118, Adres: 0x761560, Sonraki: 0x761540
10. Dugum - Veri: 436, Adres: 0x761540, Sonraki: 0
```

İlk eklenen dugum (son dugum): 436

Createlist() fonksiyonu rastgele 10 sayı üretir ve her sayı için yeni düğüm oluşturur. Düğümleri addhead kullanarak listenin başına ekler. Dumplist() çıktısıyla 10 nodeun tamamı gösterilir.

ÖDEV -3:

```
struct node {
    int data;
    node* link;
};

int ciftsay(node* list) {
    int count = 0;
    while (list != NULL) {
        if (list->data % 2 == 0) count++;
        list = list->link;
    }
    return count;
}

int main() {
    node* list = NULL;

    node* node1 = new node{100, NULL};
    node* node2 = new node{55, NULL};
    node* node3 = new node{42, NULL};

    list = node1;
    node1->link = node2;
    node2->link = node3;

    cout << "cift sayili dugum adedi: " << ciftsay(list);
}
```

cift sayili dugum adedi: 2

node ile düğüm oluştur. 3 düğüm ekledik. Ciftsay() fonksiyonuyla çiftleri kontrol et sonucu yazdırdık.

ÖDEV – 4 :

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

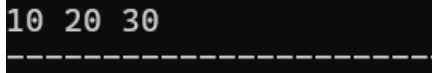
void addlast(Node*& head, int value) {
    Node* yeni = new Node{value, NULL};

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

int main() {
    Node* liste = NULL;

    addlast(liste, 10);
    addlast(liste, 20);
    addlast(liste, 30);

    Node* gecici = liste;
    while (gecici != NULL) {
        cout << gecici->data << " ";
        gecici = gecici->next;
    }
}
```



10 20 30

Yeni bir düğüm oluşturduk. Liste eğer boşsa direkt başa ekledik. Eğer liste doluysa son düğüme gidip yeni düğümü var sondaki düğüme bağlarız.

(Bu bölüme sıradaki ödev numarasını ve ödevi yazın.)

ÖDEV :