A yığınındaki sayıları B yığınına geçirecek kodu yazınız.
Preorder postorder inorder

7- Bankalardaki müşterileri kabul etme sırası düşünüldüğünde (Kredi kartı ile fiş almak gibi.) en uygun veri yapısı **Öncelikli Kuyruk** verilebilir. (5 p)

Kuyruk yazan 5 puan

Öncelikli Kuyruk yazan 10 puan

5- Aşağıda kodu verilen **ve elemanlarının küçükten büyüğe sıralı olduğu** tek yönlü bağıl listenin ekle metodunu yazınız (15 p).

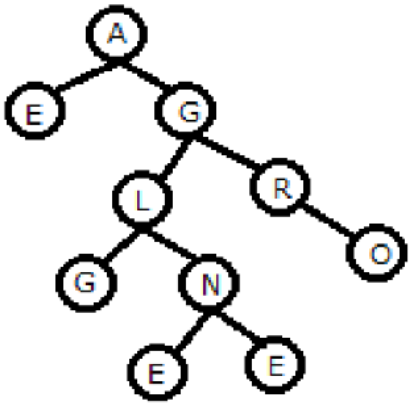
```
struct dugum
{
    int bilgi;
    dugum *sonraki;
};

class Liste
{
  private:
      dugum *ilk;
  public:
      Liste()
      {
          ilk=NULL;
      }
      void yaz()
      {
          dugum *gecici=ilk;
          while(gecici!=NULL)
              {
           cout<<"Deger:"<<gecici->bilgi<<endl;
                  gecici=gecici->sonraki;
              }
      }
```

```
void ekle(int deger)
 {
        dugum *yeni=new dugum;
          yeni->bilgi=deger;
          yeni->sonraki=NULL;
          if(ilk==NULL) ilk=yeni;
          else
            {
              dugum *gecici=ilk;
              dugum *onceki=NULL;
              while(gecici!=NULL&&yeni->bilgi>gecici->bilgi)
              {
                  onceki=gecici;
                  gecici=gecici->sonraki;
              }
              if(onceki==NULL)
                      {
                        yeni->sonraki=ilk;
                        ilk=yeni;
                      }
              else {
                  onceki->sonraki=yeni;
                  yeni->sonraki=gecici;
                  }
            }

  }
}; // Liste sınıfının sonu
```

2- Aşağıdaki İkili ağacı preorder, inorder ve postorder şeklinde dolaşınız (15 p).



PREORDER: A E G L G N E E R O

INORDER: E A G L E N E G R O

POSTORDER: E G E E N L O R G A

```
int k=0;
int kattoplam(int A[], int i, int n){
  if (n == 1) return A[i];
  else{
      int toplam=kattoplam( A, i + n/2, n/2)+kattoplam( A, i , n/2 )
      cout<<++k<<". toplam:"<<toplam<<endl;
      return toplam;
   }
}
int main() {
       int A[8]={8,7,6,5,4,3,2,1};
       kattoplam(A,0,8);
      return 0;
}
```

1. Yandaki program kodunun ekran çıktısını aşağıdaki uygun yerlere yazınız? (Ö.Ç. 1,2 P.Ç. 1,3) (**14p**)

1. toplam:
2. toplam:
3. toplam:
4. toplam:
5. toplam:
6. toplam:
7. toplam:

2. Aşağıda istenen algoritmaları bırakılan boşluklara yazınız? (**20p**)

a. Çift yönlü bağıl listede araya eleman ekleme işlemi (C++)

b. Tek yönlü bağıl listede ilk elemanı çıkarma işlemi (C++)

4. Aşağıda Dizi ile gerçekleştirilmiş Yığıt kodunu, tek yönlü bağıl liste ile gerçekleştirilmiş halini sağ tarafına yazınız. Sadece push metodu yazılacaktır? (Ö.Ç. 4,5 P.Ç. 1,2,3) (**20p**)

```
void push(const Nesne &eleman){
        if(dolumu()) yerAc(max(1,2*kapasite));
        stackBasi++;
        elemanlar[stackBasi] = eleman;
        elemanSayisi++;
}
```

5. **10-(2+6)/(10-6)+2** ifadesini postfix'e yığıt kullanarak dönüştürünüz, postfix ifadenin sonucunu yine yığıt kullanarak hesaplayınız. İşlemler adım adım gösterilmelidir. **Bu soruyu sayfanın arkasına çözünüz**. (Ö.Ç. 5 P.Ç. 2,3) (**19p**)
6. Algoritma karmaşıklığı ne demektir kısaca açıklayınız? (**6p**)

1- Aşağıdaki a ve b'de istenen algoritma sözde kod şeklinde olacağı için C++ kullanımı zorunlu değildir. (20 p)

a. İki yönlü bağıl listede araya eleman eklemek için bir algoritma (sözde kod) yazınız.

b. Dairesel kuyruğa eleman ekleme algoritmasını (sözde kod) yazınız.

a.

```
void insert(eleman, konum){
        ListeGezici *itr = oncekiniKonumuileBul(konum);
        Dugum *onceki = itr.simdiki->ileri;
        itr. simdiki ->ileri = new Dugum(eleman, itr. simdiki ->ileri);

        if(onceki != NULL) önceki->geri = itr. simdiki ->ileri;
}
```

b.

```
void enqueue(eleman) {

        if(isEmpty()){
                on = arka = new Dugum(eleman);
                arka->ileri = on;
        }
        else{
                Dugum *eskiArka = arka;
                arka = new Dugum(eleman, on);
                eskiArka->ileri = arka;
        }
}
```

6- Bir bankada müşterilerin yatırdıkları paralarına faiz uygulanmaktadır. Yatırılan paranın her iki ayda bir faiz artışı gerçekleştirilmektedir. Parasını yatırdığı ay tek sayı ise ilk ay faizi 0,02, eğer çift sayı ise 0,01 dir. Yatırılan para her iki ayda bir 0,01 artırılmaktadır. Paranın miktarını ve yatırım ay sayısı öğrenildikten sonra toplam birikimi hesaplayan programı rekürsif (özyinelemeli) fonksiyon kullanarak gerçekleyiniz. (15 p)

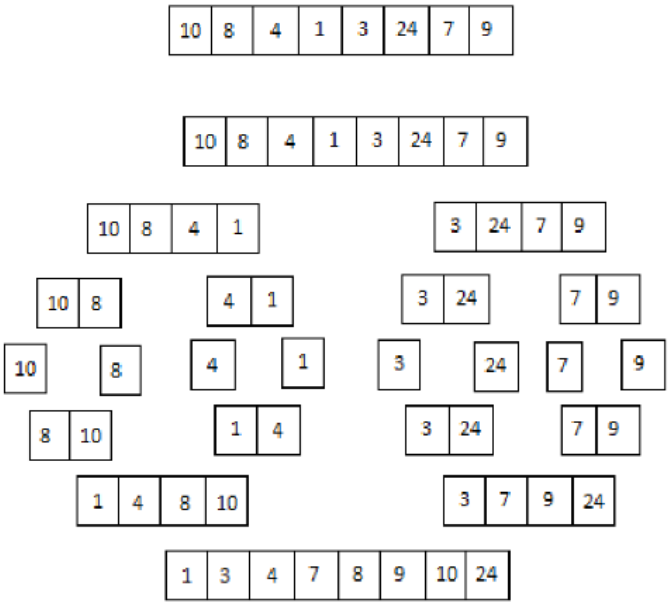
Örnek:
100 TL:
1 ay için:100+100*0.02=102
3 ay için:
100+100*0.02=102
102+102*0.01=103.02
5 ay için:
100+100*0.02=102
102+102*0.01=103.02
103.02+103.02*0.01=104.05
2 ay için :100+100*0.01=101
4 ay için: 100+100*0.01=101
101+101*0.01=102.01
6 ay için: 100+100*0.01=101
101+101*0.01=102.01
102.01+102.01*0.01=103.03

```
double faiz(int miktar,int ay)
{
  if(ay==1) return miktar*1.02;
  if(ay==2) return miktar*1.01;
  else return (faiz(miktar,ay-2)+faiz(miktar,ay-2)*0.01);
}
```

1-) **Dizi ile gerçekleştirilmemiş** İkili arama ağacında düğüm sayısını bulan fonksiyonu yazınız. (15 p)

```
int BinaryTree::Uzunluk(Node *alt_root) const{
        if(alt_root == NULL) return 0;
        else return (Uzunluk(alt_root->sol) + 1 + Uzunluk(alt_root->sag));
}
int BinaryTree::Uzunluk() const{
        return Uzunluk(root);
}
```

3.) Aşağıdaki diziyi Merge Sort kullanarak sıralayınız. Sıralama işleminin her adımı çizilerek gösterilmelidir. (10p)



4.) Öz yinelemeli fonksiyonlar yığın kullanılarak döngüsel hale getirilebilirler. Örneğin faktoriyel hesabı yapan ***f*** fonksiyonunun döngüsel hali ***f_stack*** fonksiyonudur. Buna göre fibonacci sayılarını hesaplayan ***Fib*** fonksiyonunun döngüsel fonksiyon (Stack kullanan) halini yazınız. (15 p)

```
int f(int sayi) {
        if(sayi==1) return 1;
        return sayi*f(sayi-1);
}
```

```
int f_stack(int sayi) {
        Stack<int> s;
        while(sayi) s.push(sayi--);
        int sonuc =1;
        while(!s.bosmu(){
                sonuc*=s.pop();
        }
        return sonuc;
}
```

```
int Fib(int sayi)
{
        stack yigin;
        if(sayi==0)
                return 0;

        yigin.push(0);
        yigin.push(1);

        while((--sayi)>0)
        {
                int pop1 = yigin.pop();
                int pop2 = yigin.pop();
                yigin.push(pop1);
                yigin.push(pop1+pop2);
        }
        return yigin.pop();
}
```

6.) Dikdörtgen içerisinde bulunan kod parçasındaki boşlukları (toplam 5 adet) uygun bir şekilde doldurunuz. (10 p)

```
const int eLEMANsAYISI=10;//Eleman sayısı

class HashTablosu
{
    private:
        ListeYeni* depolamaBirimi[eLEMANsAYISI];
    public:
        HashTablosu();
        int hashKoduUret(const string &giris);
        void ekle(const string &giris);
        void yazdir(const string &giris);
        void tumunuYazdir();
};
```

```
void HashTablosu::.............(const string &giris)
{

    ...............[.............(giris)%.........]->basaEkle(........);
}
```

```
void HashTablosu::ekle(const string &giris)
{

   depolamaBirimi[hashKoduUret(giris)%eLEMANsAYISI]->basaEkle(giris);
}
```

```
int* DiziKopyala(int *kaynak,int uzunluk){
    int *hedef = new int[uzunluk];
    for(int i=0;i<uzunluk;i++){
        hedef[i] = kaynak[i];
    }
    return hedef;
}
```

```
bool DiziKarsilastir(int* d1,int u1,int *d2,int u2){
    if(u1 != u2) return false;
    for(int i=0;i<u1;i++){
        if(d1[i] != d2[i]) return false;
    }
    return true;
}

int main(){
    int p[] = {10,20,30};
    int r[] = {10,20,30};

    if(DiziKarsilastir(p,3,r,3)) cout<<"Diziler esit";
    else cout<<"Diziler esit degil";
    return 0;
}
```

```
void diziTersCevir(int dizi[],int uzunluk){
    if(uzunluk > 1){
        int tmp = dizi[0];
        dizi[0] = dizi[uzunluk-1];
        dizi[uzunluk-1] = tmp;

        diziTersCevir(dizi+1,uzunluk-2);
    }
}


int main(){
    int sayilar[] = {27,46,17,90,63};
    diziTersCevir(sayilar,5);
    for(int index=0;index<5;index++){
        cout<<sayilar[index]<<" ";
    }
    return 0;
}
```

```
bool ikiliArama(int sayilar[],int baslangic,int bitis,int aranan){
    int ortaindis = (baslangic+bitis)/2;
    if(bitis < baslangic) return false;
    if(sayilar[ortaindis] == aranan) return true;
    if(aranan < sayilar[ortaindis])
        return ikiliArama(sayilar,baslangic,ortaindis-1,aranan);
    else
        return ikiliArama(sayilar,ortaindis+1,bitis,aranan);
}


int main(){
    int dizi[] = {17,27,46,63,90,112,125,140,157};
    int sayi;
    cout<<"Aranan Sayi:";
    cin>>sayi;
    if(ikiliArama(dizi,0,8,sayi)) cout<<"aranan sayi var"<<endl;
    else cout<<"aranan sayi yok"<<endl;
    return 0;
```

```
Node<Object>* FindPrevByPosition(int position){
    if(position<0 || position>size) throw "Index out of range";
    int index=1;
    for(Node<Object>* itr=head;itr!=NULL;itr=itr->next,index++){
        if(position == index) return itr;
    }
    return NULL;
}
```

ASAL BULMA ÖZYİNELEME

```
#include <iostream>

bool isPrimeRecursive(int num, int i = 2)
{
    // 2'den küçük olan sayılar asal sayı değildir
    if (num < 2)
        return false;

    // 2 asal bir sayıdır veya i'nin num ile bölünmesi durumunda asal sayı olmadığını belirtir
    if (i == num)
        return true;
    if (num % i == 0)
        return false;

    // i'yi artırarak sayının asallığını kontrol et
    return isPrimeRecursive(num, i + 1);
}

int main()
{
    int num;
    std::cout << "Enter a number: ";
    std::cin >> num;
    if (isPrimeRecursive(num))
        std::cout << num << " is a prime number.\n";
    else
        std::cout << num << " is not a prime number.\n";
    return 0;
}
```

FİBONACCİ

```
#include <iostream>

int fibonacci(int n)
{
    if (n <= 1)
```

```
        return n;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}

int main()
{
    int n;
    std::cout << "Enter the position for Fibonacci series: ";
    std::cin >> n;
    std::cout << "Fibonacci number at position " << n << " is " << fibonacci(n) << std::endl;
    return 0;
}
```

FAKTÖRİYEL

```
#include <iostream>

int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}

int main()
{
    int num;
    std::cout << "Enter a number: ";
    std::cin >> num;
    std::cout << "Factorial of " << num << " is " << factorial(num) << std::endl;
    return 0;
}
```

EBOB

```
#include <iostream>

int gcd(int a, int b)
{
    if (b == 0)
        return a;
    else
        return gcd(b, a % b);
}

int main()
{
    int num1, num2;
```

```
    std::cout << "Enter two numbers: ";
    std::cin >> num1 >> num2;
    std::cout << "Greatest Common Divisor (GCD) of " << num1 << " and " << num2 << " is "
<< gcd(num1, num2) << std::endl;
    return 0;
}
```

EKOK

```
#include <iostream>

int gcd(int a, int b)
{
    if (b == 0)
        return a;
    else
        return gcd(b, a % b);
}

int lcm(int a, int b)
{
    return (a * b) / gcd(a, b);
}

int main()
{
    int num1, num2;
    std::cout << "Enter two numbers: ";
    std::cin >> num1 >> num2;
    std::cout << "Least Common Multiple (LCM) of " << num1 << " and " << num2 << " is " <<
lcm(num1, num2) << std::endl;
    return 0;
}
```