**ÇIKIN SORU ÇÖZÜMLERİ**

**Soru**: Verilen bir N sayısını çıkın kullanarak ikilik düzende ekrana yazan bir fonksiyon yazınız.

Onluk düzende yazılmış bir sayıyı ikilik düzene çevirmek için sayıyı 0 olana kadar üst üste ikiye bölmek daha sonra ise bu bölme işlemlerine kalanları ters sırada okumak gerekir.Örneğin 10 sayısını ikilik düzene çevirmek için ;

* 10 ikiye bölünür.Bölüm 5 kalan 0 dır
* 5 ikiye bölünür.Bölüm 2 kalan 1 dir
* 2 ikiye bölünür bölüm 1 kalan 0 dır
* 1 ikiye bölünür.Bölüm 0 kalan 1 dir.

Bölme işlemlerinden kalanları ters sırada okursak ,1010 elde ederiz.

1. Void ikilikDuzen(int N){
2. Int hane;
3. Cikin c;
4. Ornek e;
5. C=new Cikin(1000);
6. While(N>0){
7. Hane =N%2;
8. e=new Ornek(hane);
9. c.CikinEkle(e);
10. N=N/2;
11. }
12. While(c.ust!=-1);
13. E=c.CikinSil();
14. System.out.println(e.icerik);
15. }
16. }

**Soru**: Verilen bir çıkının en büyük elemanını döndüren fonksiyonu yazınız.Yazdığınız fonksiyon çıkındali elemanların sırasını bozmamalıdır.

Sabit bir dizi ile tanımlı bir çıkının içerdiği **eleman**ların tanımı

Public class Ornek{

Int icerik;

Public Ornek(int icerik){

This.icerik=icerik;

}

}

Çıkında o anda işlenen ornek değişkeninde (5-9)işlenen elemana kadar olan elemanların en büyüğü de buyuk değişkeninde tutulmaktadır.Her örnek,o ana kadar görünen en büyük sayı ile karşılaştırılmakta (11),eğer o örnek o ana kadar görürnen en büyük sayıdan büyükse ,en büyük sayının değeri değiştirilir(12).Çıkından elemanları bsildikten sonra geri koyabilmek için ,her silinen eleman geçici bir çıkına eklenir(6.10).Çıkının en büyük elemanı bulunduktan sonra geciçi çıkındaki tüm elelmanlar silinip (15) tekrar orjinal çıkına eklenir(16).

1. Int enBuyuk(){
2. Int buyuk;
3. Ornek **ornek**;
4. Cikin **t**=new Cikin();
5. Ornek=CikinSil();
6. t.cikinEkle(ornek);
7. Buyuk=ornek.icerik;
8. While(!cikinBos()){
9. Ornek=cikinSil();
10. t.cikinEkle(ornek);
11. If(ornek.icerik>buyuk)
12. Buyuk=ornek.icerik;
13. }
14. While(!t.cikinBos()){
15. Ornek=t.cikinSil();
16. cikinEkle(ornek);
17. }
18. Return buyuk;
19. }

**Soru** : Verilen bir karakter dizisindeki parantezlerin eşli olup olmadığı çıkın yapısını kullanarak belirleyen fonksiyonu yazınız.Örneğin parantezler sırasıyla ( ( ) ( ) ) ise eşlidir, ( ( ( ) ( ise eşli değildir.Verilen ifadede sadece ( , ) parantez işaretleri bulunmaktadır.

Int parantez(char dizi)

-------------------------

Void CikinEkle(Eleman yeni){

Yeni.ileri=ust;

Ust=yeni;

}

--------------------

Void CikinSil(){

If(!cikinBos()){

Ust--;

Return dizi[ust+1];

}

Return null;

}

Boolean CikinBosmu(){

If(ust==-1){

Return true;

Else

Return false;

}

-------------------------------------

boolean parantez (String dizi)

Cikin ck=new Cikin(100);

For(int i=0;i<dizi.lenght();i++){

If(dizi.CharAt[i]==’ ( ‘ ){

Ck.CikinEkle(dizi.charAt[i]);

Else if(dizi.charAt[i]==’ ) ‘ ){

Ck.CikinSil();

If(ck.CikinBosmu())

Return true;

Else

Return false;

}

}

**Soru** :Sabit dizi ile tanımlanan çıkın veri yapısını çıkın dolduğunda büyütecek(çıkında tanımlanmış dizi kapasitesini iki katına artıracak ) fonksiyonu yazınız.

Void Cikinbuyut()

Void ckin\_buyut(Cikinptr c)

Algorithms push(o)

If t=S.lenght-1 then

A<--new array of

Size..

For i<--0 to t do

A[i]<--S[i]

S<--A

T<--t+1

S[t]<--0

**Soru** : Verilen bir karakter dizisinin palindrom olup olmadığını çıkın yapısını kullanarak belirleyen fonksiyonu yazınız.

// Java implementation of the approach

class GFG

{

static char []stack; //stack adında dizi

static int top = -1; //stack boş

// push function

static void push(char ele)

{

stack[++top] = ele; //ust uste ekliyo işte

}

// pop function

static char pop()

{

return stack[top--]; //çıkart

}

// Function that returns 1

// if str is a palindrome

static int isPalindrome(char str[])

{

int length = str.length;

// Allocating the memory for the stack

stack = new char[length \* 4];

// Finding the mid

int i, mid = length / 2; //küçük olana tamamlıyo küsüratlı sayıyı

for (i = 0; i < mid; i++)

{

push(str[i]);

}

// Checking if the length of the String

// is odd, if odd then neglect the

// middle character

if (length % 2 != 0)

{

i++; //uzunluğu tek ise +1

}

// While not the end of the String

while (i < length)

{

char ele = pop();

// If the characters differ then the

// given String is not a palindrome

if (ele != str[i])

return 0;

i++;

}

return 1;

}

// Driver code

public static void main(String[] args)

{

char str[] = "kabak".toCharArray();

if (isPalindrome(str) == 1)

{

System.out.printf("Yes");

}

else

{

System.out.printf("No");

}

}

}

// This code is contributed by PrinciRaj1992

}

}

**Soru**: Java'da hazır Stack (yığıt) sınıfı da bulunmaktadır. Aşağıdaki örnekte String'ler, oluşturulan s yığıtına yerleştirilerek ters sırada listelenmektedir.



public static void main(String args[]) 

{

String str[] = { "Bilgisayar", "Dolap", "Masa", "Sandalye", "Sıra" }; 

Stack s = new Stack(); 

for(int i=0; i < str.length; i++)

s.push(str[i]); 

while(!s.empty() )

System.out.println(s.pop());  }  }

. 

**Soru** : Sabit dizi ile tanımlanan bir çıkının en üstündeki iki elemanı yer değiştiren fonksiyonu yazınız.

Void yerdegistir(Cikin s){

int üstteki=s.ust;

s.pop();

int yeniüst=s.ust;

s.pop();

s.push(ustteki);

s.push(yeniust);

}

**Soru** : Verilen bir çıkının en altındaki elemanı döndüren fonksiyonu yazınız.Yazdıgınız fonksiyonda sadece çıkına ekleme,çıkından silme ve çıkının boş oldugunu kontrol etme fonksiyonları kullanabilirisiniz.

eleman enalttakinibul(cikin s){

eleman cikaaneleman;

while(!s.bosmu()){

cikaaneleman=s.pop();

}

return cikaaneleman

}

**Soru** : Tek bir sabit dizi kullanarak iki çıkını birlikte tanımlamak mümkün olabilir.Birinci çıkın indisi 0 olan elemandan başlayıp yeni elemanlar artan indislere yerleştirilecek şekilde büyürken, ikinci çıkın indisi N -1 olan elemandan başlayıp yeni elemanlar azalan indislere yerleştirilecek şekilde büyüyebilr.Bu amacı gercekleştirecek çıkın veri yapısını tanımlayınız.

// Java program to implement two stacks in a

// single array

class TwoStacks {

int size;

int top1, top2;

int arr[];

// Constructor

TwoStacks(int n)

{

arr = new int[n];

size = n;

top1 = -1;

top2 = size;

}

// Method to push an element x to stack1

void push1(int x)

{

// There is at least one empty space for

// new element

if (top1 < top2 - 1) {

top1++;

arr[top1] = x;

}

else {

System.out.println("Stack Overflow");

System.exit(1);

}

}

// Method to push an element x to stack2

void push2(int x)

{

// There is at least one empty space for

// new element

if (top1 < top2 - 1) {

top2--;

arr[top2] = x;

}

else {

System.out.println("Stack Overflow");

System.exit(1);

}

}

// Method to pop an element from first stack

int pop1()

{

if (top1 >= 0) {

int x = arr[top1];

top1--;

return x;

}

else {

System.out.println("Stack Underflow");

System.exit(1);

}

return 0;

}

// Method to pop an element from second stack

int pop2()

{

if (top2 < size) {

int x = arr[top2];

top2++;

return x;

}

else {

System.out.println("Stack Underflow");

System.exit(1);

}

return 0;

}

// Driver program to test twoStack class

public static void main(String args[])

{

TwoStacks ts = new TwoStacks(5);

ts.push1(5);

ts.push2(10);

ts.push2(15);

ts.push1(11);

ts.push2(7);

System.out.println("Popped element from"

+ " stack1 is " + ts.pop1());

ts.push2(40);

System.out.println("Popped element from"

+ " stack2 is " + ts.pop2());

}

}

// This code has been contributed by

// Amit Khandelwal(Amit Khandelwal 1).

**KUYRUK SORU ÇÖZÜMLERİ**

**Soru** : Sabit dizi ile tanımlanan bir kuyrugun en arkasındaki elemanı kuyrugun en önüne getiren fonksiyonu yazınız.

Önce kuyrugun önünde yer ayarlanır bir diğer deyişle kuyrugun başını gösteren işaretci bir azaltılır.Kuyrugun sonunu gösteren işaretci bir azaltılır.Son olarak kuyrugun başına kuyrugun sonundaki eleman getirilir.

Void oneGetir(){

Bas=(bas-1)%N;

Son=(s0n-1)%N;

Dizi[bas]=dizi[son];

}

**Soru :** Kuyruk tanımını ve kuyruk fonksiyonları kuyruga\_ekle vekuyruk\_sil i çıkın kullanrak tanımlayınız.Yazdığınız fonksiyonları karmaşıklık analizini yapınız.

Public class Kuyruk{

Cikin dizi;

Void Kuyruk(int N){

Kuyruk baştan çıkışlı çıkın ise sondan çıkışlı veri yapılarıdır.Yeni kuyruk tanımından eleman silmek için kuyruga ait çıkının ilk elemanını silmek gerekir.

Dizi=new Cikin();

}

---------------------------------------

Ornek kuyrukSil(){

Ornek e,sonuc;

Cikin c;

If(!dizi.cikinBos())}

C=Cikin(N);

While(!dizi.cikinBos()){

E=dizi.cikinSil();

c.cikinEkle(e);

}

Sonuc=c.cikinSil(9;

While(!c.cikinBos()){

E=c.cikinSil();

Dizi.cikinEkle(e);

}

Return sonuc;

}

}

------------------------------------------

Void kuyrugaEkle(Ornek yeni){

If(!dizi.cikinDolu())

Dizi.cikinEkle(yeni);

}

-------------------------------------------

**Soru** : Sabit dizi ile tanımlanan bir kuyrugun en önündeki elemanı kuyrugun en arkasına götüren fonksiyonu yazınız.

void basisonaat(eleman []k){

eleman silinen=k.kuyruktansil();

k.kuyrugaekle(silinen);

}

**Soru** : Bağlı liste ile tanımlanan bir kuyrugun en önündeki elemanı kuyrugun en arkasına götüren fonksiyonu yazınız.

----------------

Public class Kuyruk{

Eleman bas;

Eleman son;

Public Kuyruk(){

Bas=null;

Son=null;

}

-----------------------

Void kuyrukEkle(Eleman yeni){

If(!kuyrukBos()){

Som.ileri=yeni;

Else

Bas=yeni;

Son=yeni;

}

----------------------------

Void kuyrukSil(){

Eleman sonuc;

Sonuc=bas;

If(!kuyrukBos()){

Bas=bas.ileri;

If(bas==null)

Son=null;

}

Return sonuc;

}

---------------

Void basıSonaAt(Eleman yeni){

Eleman silinen=yeni.kuyrukSil();

Yeni.kuyrukEkle(silinen);

}

------------------------

**Soru**: kuyruk veri yapısını kuyruk doldugunda genişleyecek (kuyrukta tanımlanmış dizininkapasitesini iki katına artıracak) şekilde yeniden tanımlayınız.

void eskiyiyeniyetasi(ArrayQueque eskikuyruk, ArrayQueque yenikuyruk) {

while(!eskikuyruk.kuyrukBos()){

eleman temp=eskikuyruk.kuyruksil();

yenikuyruk.kuyrugaekle(temp);

}

}

Public static void main (String [] args]{

ArrayQueque eskikuyruk = new ArrayQueque(200)

if (eskikuyruk.kuyrukDolu() ) {

ArrayQueque yenikuyruk = new ArrayQueque(eskikuyruk.capacity \* 2);

eskiyiyeniyetasi(eskikuyruk, yenikuyruk);

}

Boolean kuyrukDolu(){

If(bas==(son+1)%N)

Return true;

Else

Return false;

}

**Soru** :Bu bölümde dizi ile yapılan kuyruk tanımında dizinin tüm elemanları değil ancak dizideki elelamların bir eksiği kadar kullanılabilmektedir.Dizideki tüm elemanları kullanabiilecek şekilde kuyruk tanımını değiştiriniz.

Public class Kuyruk{

Ornek dizi[];

Int bas;

Int son;

Int N;

Public Kuyruk(int N){

Dizi=new Ornek[N];

This.N=N;

Bas=0;

Son=0;

}

**Soru** : Verilen bir kuyrugun en büyük elemanı döndüren fonksiyonu yazınız.Yazdığınız fonksiyon kuyruktaki lelemanların sırasını bozmamalıdır ve sadece kuyruktaki temel fonksiyonlar kullanılablir.

eleman enbuyukbul(){

Queque gecicikuyrugumuz=new Queque();

eleman temp;

Eleman enbuyuk=0;

while(!kuyrugumuz.isEmpty()){

temp=kuyrugumuz.kuyruktansil();

gecicikuyrugumuz.kuyrugaekle(temp)

if (temp.deger>enbuyuk.deger) {

enbuyuk=temp;

}

}

while(!gecicikuyrugumuz.isEmpty()){

temp=gecicikuyrugumuz.kuyruktansil();

kuyrugumuz.kuyrugaekle(temp);

}

return enbuyuk

}

Psvm(){

Queque kuyrugumuz=new Queque();

Eleman civciv= kuyrugumuz.enbuyukbul();

System.out.println(“değer = “+ civciv.deger);

}

**Soru**: bağlı liste tanımlanan bir kuyrugun en arkasındaki elemanı kuyrugun en önüne getiren fonksiyonu yazınız.

Void oneGetir(){

Eleman ara=bas;

Bas=son;

Son=ara;

}

**Soru**:Sabit dizi ile tanımlanan bir kuyrugun ilk elemanının arkasına yeni bir eleman ekleyen fonksiyonu yazınız.

Void oneGetir() {

**Soru**:Bağlı liste ile tanımlanan bir kuyrugun ilk elemanını arkasına yeni bir eleman ekleyen fonksiyonu yazınız.

Public class Kuyruk{

Eleman bas;

Eleman son;

Public Kuyruk(){

Bas=null;

Son=null;

}

Void ilkEkle(Eleman yeni){

If(!kuyrukbos()){

Yeni.ileri=bas.ileri;

Bas.ileri=yeni;

}

**Soru** : Verilen bir kuyrugun büyüklüğünü Mye düşüren fonksiyonu yazınız.Kuyrukta büyüklüğü azaltılmadan önce en fazla M-1 eleman olduğunu varsayılanız.

**Soru**:Her iki yönden de (bas ve son) eklmem ve silme yapılabilen kuyruk veri yaoısını çift bağlı liste yapısını kullanarak tanımlayınız.