Veri Yapıları ile İlgili Sorular (Tamamen ChatGPT)

Doğru/Yanlış İfadeleri

1. **Dizi veri yapıları, sabit boyutlu ve ardışık bellek alanında depolanır.**
2. **Yığın (stack) veri yapısında, elemanlar son giren ilk çıkar (LIFO) prensibine göre işlenir.**
3. **Bağlı listelerde, her düğüm yalnızca bir veri alanı içerir.**
4. **Kuyruk (queue) veri yapısı, ilk giren ilk çıkar (FIFO) prensibine göre çalışır.**
5. **Hash tablosunda çakışma meydana geldiğinde, zincirleme yöntemi kullanılabilir.**
6. **Bir bağlı liste, elemanların rastgele erişimine izin verir.**
7. **Yığın veri yapısında "push" işlemi, en üstteki elemanı yığının dışına çıkarır.**
8. **Dizi ve bağlı liste arasında en önemli fark, dizilerin boyutunun sabit olmasıdır.**
9. **Bir bağlı liste, eleman ekleme ve silme işlemlerinde dizilere göre daha hızlıdır.**
10. **Yığın (stack) veri yapısında, en üstteki elemanı silmek için "push" işlemi kullanılır.**
11. **Kuyruk (queue) veri yapısında, elemanlar yalnızca arka taraftan eklenebilir ve ön taraftan silinebilir.**
12. **Hash tablosunda, anahtarlar benzersiz olmalıdır.**
13. **Diziler, bellek alanında ardışık olarak depolanmadıkları takdirde verimli bir şekilde çalışamazlar.**
14. **Bir yığın veri yapısında, "peek" işlemi yığının en üstündeki elemanı silmeden görüntüler.**
15. **Bağlı listeler, elemanların sıralı bir şekilde depolanmasını zorunlu kılar.**
16. **Hash tablosu kullanılarak yapılan arama işlemleri, genellikle O(1) zaman karmaşıklığına sahiptir.**

Soru 1:

Aşağıdakilerden hangisi bir veri yapısıdır?

A) Fonksiyon
B) Değişken
C) Dizi
D) Operatör

Soru 2:

Ağaç veri yapısında, her düğümün en fazla kaç çocuğu olabilir?

A) 1
B) 2
C) 3
D) Sınırsız

Soru 3:

Aşağıdakilerden hangisi yığın (stack) veri yapısının temel özelliklerinden biridir?

A) İlk giren, ilk çıkar (FIFO)
B) Son giren, ilk çıkar (LIFO)
C) Rastgele erişim
D) Dizi tabanlı sıralama

Soru 4:

Bağlı liste (linked list) veri yapısında her düğümün ne tür bir bilgi taşıması beklenir?

A) Sadece sayılar
B) Sadece karakterler
C) Veri ve bir sonraki düğümün adresi
D) Sadece adres bilgisi

Soru 5:

Hangi veri yapısı, verileri sıralı bir şekilde depolamak için en uygun olanıdır?

A) Yığın (Stack)
B) Kuyruk (Queue)
C) Ağaç (Tree)
D) Hash Tablosu

Soru 6:

Hangi veri yapısı, son giren ilk çıkar (LIFO) prensibine göre çalışır?

A) Dizi
B) Kuyruk
C) Yığın (Stack)
D) Bağlı Liste

Soru 7:

Aşağıdakilerden hangisi bağlı liste (linked list) türlerinden biri değildir?

A) Tek yönlü bağlı liste
B) Çift yönlü bağlı liste
C) Dairesel bağlı liste
D) Matris bağlı liste

Soru 8:

Bir kuyruk (queue) veri yapısında, hangi işlem en hızlı şekilde gerçekleştirilir?

A) Ekleme (enqueue)
B) Silme (dequeue)
C) Arama
D) Güncelleme

Soru 9:

Aşağıdaki veri yapılarından hangisi, elemanların sıralı bir şekilde saklanmasını garanti etmez?

A) Dizi
B) Çift yönlü bağlı liste
C) Yığın (Stack)
D) Kuyruk

Soru 10:

Bir yığın veri yapısında, "peek" işlemi neyi ifade eder?

A) Yığının boş olup olmadığını kontrol etmek
B) Yığının en üstündeki elemanı görüntülemek, fakat silmemek
C) Yığının tüm elemanlarını silmek
D) Yığının en altındaki elemanı görüntülemek

Soru 11:

Bir bağlı listeyi tersine çevirmek için en etkili yöntem hangisidir?

A) Yeni bir bağlı liste oluşturmak
B) Düğümleri sıralamak
C) Düğümlerin bağlantılarını değiştirmek
D) Düğümleri bir diziye kopyalamak

Soru 12:

Bir hash tablosunda çakışma meydana geldiğinde, aşağıdaki yöntemlerden hangisi çakışmayı çözmek için kullanılmaz?

A) Açık adresleme
B) Zincirleme (chaining)
C) Düğüm sıralama
D) Kapsama (bucketization)

Soru 13:

Bir yığın veri yapısında "pop" işlemi gerçekleştirildiğinde hangi durum gerçekleşmez?

A) Yığının en üstündeki eleman silinir.
B) Yığın boşsa hata oluşur.
C) Yığının boyutu bir azalır.
D) En alt eleman erişilebilir hale gelir.

Soru 1:

Alttaki kodda ne yanlış var?

```c
#include <stdio.h>

int main() {
    int dizi[5] = {1, 2, 3, 4, 5};
    int toplam = 0;
    for (int i = 0; i <= 5; i++) {
        toplam += dizi[i];
    }
    printf("%d", toplam);
    return 0;
}
```

Soru 2:

Alttaki kodun çıktısı nedir?

```c
#include <stdio.h>

int main() {
    int stack[5], top = -1;
    stack[++top] = 10;
    stack[++top] = 20;
    printf("eleman: %d, index: %d", stack[top--], top);
    return 0;
}
```

Soru 3:

Alttaki kodu iki yönlü dairesel bağlı liste yapmak için neyi değiştirmeliyim?

```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int veri;
    struct Node* sonraki;
};

int main() {
    struct Node* ilk = (struct Node*)malloc(sizeof(struct Node));
    ilk->veri = 1;
    ilk->sonraki = NULL;
    return 0;
}
```

Doğru/Yanlış Cevaplar

1. Doğru - Diziler sabit boyutlu ve ardışık bellek alanında depolanır.
2. Doğru - Yığınlar LIFO prensibine göre çalışır.
3. Yanlış - Her düğüm veri ve bir sonraki düğümün adresini içerir.
4. Doğru - Kuyruklar FIFO prensibine göre çalışır.
5. Doğru - Çakışma durumunda zincirleme yöntemi kullanılabilir.
6. Yanlış - Bağlı listelerde elemanlara sıralı olarak erişilir.
7. Yanlış - "Push" işlemi yığına eleman ekler; en üstteki elemanı silmek için "pop" kullanılır.
8. Doğru - Dizilerin boyutu sabittir; bağlı listeler dinamik boyutlandırma sağlar.
9. Doğru - Bağlı listelerde ekleme ve silme işlemleri genellikle daha hızlıdır.
10. Yanlış - En üstteki elemanı silmek için "pop" işlemi kullanılır.
11. Doğru - Elemanlar arka taraftan eklenir ve ön taraftan silinir.
12. Doğru - Hash tablosunda anahtarlar benzersiz olmalıdır.
13. Yanlış - Diziler ardışık bellek alanı gerektirir; bu nedenle verimli çalışabilmeleri için ardışık depolanmalıdır.
14. Doğru - "Peek" işlemi en üstteki elemanı silmeden görüntüler.
15. Yanlış - Bağlı listelerde elemanların sıralı olması gerekmez.
16. Doğru - Hash tablolarında arama işlemleri genellikle O(1) zaman karmaşıklığına sahiptir.

Test Cevaplar

1. C
2. B
3. B
4. C
5. C
6. C
7. D
8. A
9. C
10. B
11. C
12. C
13. D

Boşluk Doldurma Cevaplar

1. Cevap: Döngü koşulu i <= 5 yerine i < 5 olmalıdır. Aksi takdirde dizinin sınırları dışına çıkılır.
   Açıklama: Dizi elemanlarına erişim 0'dan başlar ve 4'te sona erer. i <= 5 ifadesi, dizinin sınırlarının dışına çıkmaya neden olur.
2. Cevap: "eleman: 20, index: 0" (Son eklenen eleman yığından çıkarılır.)
   Açıklama: Yığın (stack) yapısında son eklenen eleman ilk çıkar (LIFO) prensibine göre çalışır. Bu nedenle 20 değeri yazdırılır.
3. Cevap: struct Node tanımına bir struct Node* onceki; alanı eklenmelidir, ve ilk'teki sonraki ve onceki kendisine eşitlenmeli.
   Kod:

```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    struct Node* onceki;
    int veri;
    struct Node* sonraki;
};

int main() {
    struct Node* ilk = (struct Node*)malloc(sizeof(struct Node));
    ilk->veri = 1;
    ilk->sonraki = ilk;
    ilk->onceki = ilk;
    return 0;
}
```

Açıklama: İki yönlü bağlı liste oluşturmak için her düğümde bir önceki düğümü işaret eden bir göstericiye ihtiyaç vardır.