ARRAY EXERCISES

DİZİ ÖRNEK SORULARI

1) Write and test the following function that returns the minimum value among the first `n` elements of the given array:
Verilen dizinin ilk n elemanı arasında minimum değeri döndüren aşağıdaki fonksiyonu yazıp test edin:

```
float min(float a[],i nt n);
```

2)Write and test the following function that returns the index of the first minimum value among the first `n` elements of the given array:
Belirtilen dizinin ilk n öğeleri arasında ilk minimum değerin indeksini döndüren aşağıdaki fonksiyonu yazıp test edin:

```
int minIndex(float a[],int n);
```

3) Write and test the following function that returns through its reference parameters both the maximum and the minimum values stored in an array:
Bir dizide saklanan maksimum ve minimum değerleri referans parametreleriyle döndüren aşağıdaki fonksiyonu yazın ve test edin:

```
void getExtremes(float& min,float& max,float a[],int n);
```

4) Write and test the following function that returns through its reference parameters both the largest and the second largest values (possibly equal) stored in an array:
Bir dizide saklanan hem en büyük hem de ikinci en büyük değerleri referans parametreleriyle döndüren aşağıdaki fonksiyonu yazın ve test edin:

```
void largest(float& max1,float& max2,float a[],int n);
```

5) Write and test the following function that removes an item from an array:

```
void remove(float a[],int& n, int i);
```

The function removes `a[i]` by shifting all the elements above that position are down and decrementing `n`.
Bir elemanı diziden kaldıran fonksiyonu yazıp test edin. Fonksiyon, yukarıdaki tüm elemanları aşağıya kaydırıp n değerini azaltarak bir a[i] 'yi kaldırır.

6) Write and test the following function that attempts to remove an item from an array:

```
bool removeFirst(float a[],int& n,float x);
```

The function searches the first `n` elements of the array `a` for the item `x`. If `x` is found, its first occurrence is removed, all the elements above that position are shifted down, `n` is decremented, and **`true`** is returned to indicate a successful removal. If `x` is not found, the array is left unchanged and **`false`** is returned.

Fonksiyon a[] dizisinin ilk n elemanı içinde x öğesini arar. Eğer x bulunursa, ilk karşılaşılan x çıkarılır, bu konumun üzerindeki tüm öğeler aşağı kaydırılır, n azalır ve başarılı bir ortadan kaldırmayı belirtmek için true döndürülür. X bulunmazsa, dizi değişmeden bırakılır ve false döndürülür.

7) Write and test the following function that removes items from an array:

```
void removeAll(float a[],int& n,float x);
```

The function removes all occurrences of `x` among the first `n` elements of the array `a` and decreases the value of `n` by the number removed.
a[] dizisinin ilk n öğeleri arasındaki tüm x oluşumlarını kaldıran ve kaldırılan sayı kadar n değerini azaltan fonksiyonu yazıp test edin.

8) Write and test the following function:

```
void rotate(int a[],int n,int k);
```

The function "rotates" the first `n` elements of the array `a`, `k` positions to the right (or `-k` positions to the left if `k` is negative). The last `k` elements are "wrapped" around to the beginning of the array. For example, the call `rotate(a,8,3)` would transform the array {22,33,44,55,66,77,88,99} into {77,88,99,22,33,44,55,66}. The call `rotate(a,8,-5)` would have the same effect.
a[] dizisinin ilk n elemanını k pozisyonu kadar sağa çeviren fonksiyonu yazıp test edin.

9) Write and test the following function:

```
void append(int a[],int m,int b[],int n);
```

The function appends the first `n` elements of the array `b` onto the end of the first `m` elements of the array `a`. It assumes that `a` has room for at least `m + n` elements. For example, if `a` is {22,33,44,55,66,77,88,99} and `b` is {20,30,40,50,60,70,80,90} then the call `append(a,5,b,3)` would transform `a` into {22,33,44,55,66,20,30,40}. Note that `b` is left unchanged and only `n` elements of `a` are changed.
Fonksiyon b dizisinin baştan ilk n elemanlarını a dizisinin sondan m. elemanının sonuna ekleyen fonksiyonu yazıp test edin.

10) Write and test the function

```
void insert(float a[],int& n, float x)
```

This function inserts the item `x` into the sorted array `a` of `n` elements and increments `n`. The new item is inserted at the location that maintains the sorted order of the array. This requires shifting elements forward to make room for the new `x`. (Note that this requires the array to have at least `n`+1 elements allocated.)

Sıralı ve n elemalı a dizisine x elemanını ekleyen ve n eleman sayısını artıran fonksiyonu yazıp test edin. Yeni öğe, dizinin sıralanmış sırasını koruyan konuma eklenir. Bu, yeni x'e yer açmak için elemanların ileri kaydırılmasını gerektirir. (Bunun dizinin ayrılmış en az n + 1 öğeye sahip olmasını gerektirdiğini unutmayın.)

11) Write and test the function

```
int frequency(float a[],int n,int x);
```

This function counts the number of times the item `x` appears among the first `n` elements of the array `a` and returns that count as the frequency of `x` in `a`.

X elemanının a[] dizisinin ilk n elemanı içerisindeki görülme sıklığını sayan ve bu değeri döndüren fonksiyonu yazıp test edin.

12) Write and test the following function:

```
void reverse(int a[],int n);
```

The function reverses the first `n` elements of the array. For example, the call `reverse(a,5)` would transform the array {22,33,44,55,66,77,88,99} into {66,55,44,33,22,77,88,99}.

Bir dizinin ilk n elemanlarını tersine çeviren fonksiyonu yazıp test edin.

13) Write and test the following function:

```
void add(float a[],int n,float b[]);
```

The function adds the first `n` elements of `b` to the corresponding first `n` elements of `a`. For example, if `a` is {2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9} and `b` is {6.0,5.0,4.0,3.0,2.0,1.0}, then the call `add(a,5,b)` would transform `a` into {8.2,8.3,8.4,8.5,8.6,7.7,8.8,9.9}.

b[] dizisinin nin ilk n elemanlarını, a[] dizisine karşılık gelen ilk n elemanlarına ekleyen fonksiyonu yazıp test edin.

14) Write and test the following function:

```
float innerProduct(float a[],int n,float b[]);
```

The function returns the *inner product* (also called the "dot product" or "scalar product") of the first `n` elements of `a` with the first `n` elements of `b`. This is defined as the sum of the products of corresponding terms. For example, if `a` is the array {2.2, 3.3,4.4, 5.5,6.6,7.7,8.8,9.9} and `b` is the array {4.0,–3.0,2.0,–1.0,0.0,0.0}, then the call `innerProduct(a,5,b)` would return (2.2)(4.0) + (3.3)(–3.0) + (2.2)(4.0) + (5.5)(–1.0) + (6.6)(0.0) = 2.2.

a[] dizisinin ilk n elemanı ile b[] dizisinin ilk n elemanını nokta-çarpım şeklinde çarpan ve bu değeri döndüren fonksiyonu yazıp test edin.

15) Write and test a function that implements the *Perfect Shuffle* of a one-dimensional array with an even number of elements. For example, it would replace the array `{11,22,33,44,55,66,77,88}` with the array `{11,55,22,66,33,77,44,88}`.

Tek boyutlu bir dizinin Mükemmel Karıştırmasını uygulayan bir fonksiyon yazın ve test edin.

16) Write and test the following function:

```
bool isSymmetric(int a[],int n);
```

The function returns true if and only if the array obtained by reversing the first `n` elements is the same as the original array. For example, if `a` is {22,33,44,55,44,33,22} then the call `isSymmetric(a,7)` would return **`true`**, but the call `isSymmetric(a,4)` would return **`false`**. Warning: The function should leave the array unchanged.

Bir dizinin simetrik olup olmadığını bulan fonksiyonu yazın. Fonksiyon, yalnızca ilk n öğesinin tersine çevrilmesiyle elde edilen dizi orijinal diziyle aynıysa, true döndürür.

17) Write and test the function that "rotates" 90° clockwise a two-dimensional square array of `int`s. For example, it would transform the array

11 22 33
44 55 66
77 88 99

into the array

77 44 11
88 55 22
99 66 33

İki boyutlu kare ints dizisini saatin yönünde 90 derece döndüren fonksiyonu yazın ve test edin.

18) Write and test the following function:

```
Void multiply(float a[],int n,float b[]);
```

The function multiplies the first `n` elements of `a` by the corresponding first `n` elements of `b`. For example, if `a` is the array {2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9} and `b` is the array {4.0,–3.0,2.0,–1.0,0.0, 0.0}, then the call `multiply(a,5,b)` would transform `a` into the array {8.8,–9.9,8.8,–5.5, 0.0,7.7,8.8,9.9}.
a[] dizisinin ilk n elemanlarını b[]dizisinin karşılık gelen ilk n elemanı ile çarpan fonksiyonu yazıp test edin.

19) Implement the *Insertion Sort* algorithm for sorting an array of `n` elements. In this algorithm, the main loop index `i` runs from 1 to *n*–1. On the `i`th iteration, the element `a[i]` is "inserted" into its correct position among the subarray `a[0..i]`. This is done by shifting one position up all the elements in the subarray that are greater than `a[i]`. Then `a[i]` is copied into the gap between the elements that are less than or equal to `a[i]` and those that are greater.
N elemanlı bir diziyi sırlamak için "Eklemeli Sıralama" algoritmasını uygulayın. Bu algoritmada, ana döngü indeksi i 1 ile n – 1 arasında çalışır. İ. iterasyonda, a [i] elemanı a [0..i] alt dizisi arasındaki doğru pozisyona "yerleştirilir". Bu, alt-dizideki a [i] 'den daha büyük olan tüm elemanlar bir konum yukarı kaydırılarak yapılır. Daha sonra a [i], a [i] 'den küçük veya ona eşit olan elemanlar ile daha büyük olanların arasındaki boşluğa kopyalanır.

20) Implement the *Selection Sort* algorithm for sorting an array of `n` elements. This algorithm has *n*–1 iterations, each selecting the next largest element `a[j]` and swapping it with the element that is in the position where `a[j]` should be. So on the first iteration it selects the largest of all the elements and swaps it with `a[n-1]`, and on the second iteration it selects the largest from the remaining unsorted elements `a[0..n-2]` and swaps it with `a[n-2]`, *etc*. On its *i*th iteration it selects the largest from the remaining unsorted elements `a[0..n-i]` and swaps it with `a[n-i]`.
N elemanlı bir diziyi sıralamak için "Seçmeli Sıralama" algoritmasını uygulayın. Bu algoritma, her biri bir sonraki en büyük a[j] öğesini seçip onu a[j] 'nin olması gereken konumda olan elemanla değiştiren n-1 iterasyona sahiptir. Böylece, ilk iterasyonda, tüm elemanların en büyüğünü seçer ve onu a[n-1] ile değiştirir ve ikinci iterasyonda, kalan sınıflandırılmamış elemanlardan a[0..n-2] arasından en büyüğünü seçer ve değiştirir a[n-2], vb.