Άσκηση 1: Βασική Εκθετική Αναζήτηση

Περιγραφή: Δώστε μια σειρά αριθμών και ένα στόχο, εφαρμόστε την Εκθετική Αναζήτηση για να βρείτε την θέση του στόχου στη σειρά.

Λύση:

```
using System;

class ExponentialSearchExercises
{
    static void Main()
    {
        // Άσκηση 1: Βασική Εκθετική Αναζήτηση
        int[] array1 = { 2, 4, 7, 10, 14, 20, 25, 30, 40 };
        int target1 = 25;

        int result1 = ExponentialSearch(array1, target1);
        Console.WriteLine(result1 != -1
            ? $"Ο αριθμός {target1} βρέθηκε στη θέση
{result1}."
            : $"Ο αριθμός {target1} δεν βρέθηκε.");
    }

    static int ExponentialSearch(int[] array, int target)
    {
        if (array.Length == 0)
        {
            return -1;
        }
        if (array[0] == target)
        {
            return 0;
        }

        int index = 1;
        while (index < array.Length && array[index] <= target)
        {
            index *= 2;
        }

        return BinarySearch(array, target, index / 2, Math.Min(index,
```

```
array.Length - 1));
    }

    static int BinarySearch(int[] array, int target, int left, int right)
    {
        while (left <= right)
        {
            int mid = (left + right) / 2;

            if (array[mid] == target)
            {
                return mid;
            }
            else if (array[mid] < target)
            {
                left = mid + 1;
            }
            else
            {
                right = mid - 1;
            }
        }

        return -1;
    }
}
```

Άσκηση 2: Αντικατάσταση Στόχου

Περιγραφή: Δώστε μια σειρά αριθμών και έναν στόχο που δεν υπάρχει στη σειρά. Εφαρμόστε την Εκθετική Αναζήτηση και ελέγξτε αν η συνάρτηση επιστρέφει -1 (δεν βρέθηκε).

Λύση:

```
using System;

class ExponentialSearchExercises
{
    static void Main()
    {
        // Άσκηση 2: Αντικατάσταση Στόχου
        int[] array2 = { 1, 3, 6, 8, 10, 12, 15, 18, 21 };
```

```
        int target2 = 14; // Στόχος που δεν υπάρχει

        int result2 = ExponentialSearch(array2, target2);
        Console.WriteLine(result2 != -1
            ? $"Ο αριθμός {target2} βρέθηκε στη θέση {result2}."
            : $"Ο αριθμός {target2} δεν βρέθηκε.");
    }

    static int ExponentialSearch(int[] array, int target)
    {
        if (array.Length == 0)
        {
            return -1;
        }
        if (array[0] == target)
        {
            return 0;
        }

        int index = 1;
        while (index < array.Length && array[index] <= target)
        {
            index *= 2;
        }

        return BinarySearch(array, target, index / 2, Math.Min(index, array.Length - 1));
    }

    static int BinarySearch(int[] array, int target, int left, int right)
    {
        while (left <= right)
        {
            int mid = (left + right) / 2;

            if (array[mid] == target)
            {
                return mid;
            }
```

```
            else if (array[mid] < target)
            {
                left = mid + 1;
            }
            else
            {
                right = mid - 1;
            }
        }

        return -1;
    }
}
```

Άσκηση 3: Μεγάλος Πίνακας

Περιγραφή: Εφαρμόστε την Εκθετική Αναζήτηση σε έναν μεγαλύτερο πίνακα για να βρείτε έναν συγκεκριμένο αριθμό. Ελέγξτε αν η συνάρτηση μπορεί να βρει τον αριθμό σωστά.

Λύση:

```
using System;

class ExponentialSearchExercises
{
    static void Main()
    {
        // Άσκηση 3: Μεγάλος Πίνακας
        int[] array3 = new int[100];
        for (int i = 0; i < array3.Length; i++)
        {
            array3[i] = i * 2; // Δημιουργία πίνακα με αριθμούς 0, 2, 4, 6, ...
        }
        int target3 = 84; // Στόχος σε μεγαλύτερο πίνακα

        int result3 = ExponentialSearch(array3, target3);
        Console.WriteLine(result3 != -1
            ? $"Ο αριθμός {target3} βρέθηκε στη θέση {result3}."
```

```
            : $"Ο αριθμός {target3} δεν βρέθηκε.");
    }

    static int ExponentialSearch(int[] array, int target)
    {
        if (array.Length == 0)
        {
            return -1;
        }
        if (array[0] == target)
        {
            return 0;
        }

        int index = 1;
        while (index < array.Length && array[index] <= target)
        {
            index *= 2;
        }

        return BinarySearch(array, target, index / 2, Math.Min(index,
array.Length - 1));
    }

    static int BinarySearch(int[] array, int target, int left, int right)
    {
        while (left <= right)
        {
            int mid = (left + right) / 2;

            if (array[mid] == target)
            {
                return mid;
            }
            else if (array[mid] < target)
            {
                left = mid + 1;
            }
            else
            {
```

```
                right = mid - 1;
            }
        }

        return -1;
    }
}
```

Άσκηση 4: Αναζήτηση Στην Αρχή του Πίνακα

Περιγραφή: Εφαρμόστε την Εκθετική Αναζήτηση για να βρείτε έναν αριθμό που είναι στην αρχή του πίνακα.

Λύση:

```
using System;

class ExponentialSearchExercises
{
    static void Main()
    {
        // Άσκηση 4: Αναζήτηση Στην Αρχή του Πίνακα
        int[] array4 = { 5, 10, 15, 20, 25, 30, 35, 40, 45 };
        int target4 = 5; // Στόχος στην αρχή του πίνακα

        int result4 = ExponentialSearch(array4, target4);
        Console.WriteLine(result4 != -1
            ? $"Ο αριθμός {target4} βρέθηκε στη θέση
{result4}."
            : $"Ο αριθμός {target4} δεν βρέθηκε.");
    }

    static int ExponentialSearch(int[] array, int target)
    {
        if (array.Length == 0)
        {
            return -1;
        }
        if (array[0] == target)
        {
            return 0;
```

```
        }

        int index = 1;
        while (index < array.Length && array[index] <= target)
        {
            index *= 2;
        }

        return BinarySearch(array, target, index / 2, Math.Min(index,
array.Length - 1));
    }

    static int BinarySearch(int[] array, int target, int left, int right)
    {
        while (left <= right)
        {
            int mid = (left + right) / 2;

            if (array[mid] == target)
            {
                return mid;
            }
            else if (array[mid] < target)
            {
                left = mid + 1;
            }
            else
            {
                right = mid - 1;
            }
        }

        return -1;
    }
}
```