```
private static void Swap(string[] array, int pos1, int pos2)
{
    string temp = array[pos1];
    array[pos1] = array[pos2];
    array[pos2] = temp;
}

private static string[] Bubble(string[] array)
{
    for (int i = 0; i < array.Length; i++)
    {
        for (int j = 0; j < array.Length - 1; j++)
        {
            if (String.Compare(array[j + 1], array[j]) < 0)
            {
                Swap(array, j, j + 1);
            }
        }
    }
    return array;
}

private static string[] Selection(string[] array)
{
    for (int i = 0; i < array.Length - 1; i++)
    {
        int min = i;

        for (int j = i + 1; j < array.Length; j++)
        {
            if (String.Compare(array[j], array[min]) < 0)
            {
                min = j;
            }
        }
        Swap(array, i, min);
    }

    return array;
}

private static string[] Insertion(string[] array)
{
    for (int i = 0; i < array.Length; i++)
    {
        string tmp = array[i];
        int j;

        for (j = i; j > 0; j--)
        {
            if (String.Compare(array[j - 1], tmp) < 0)
            {
                break;
            }
            array[j] = array[j - 1];
        }
        array[j] = tmp;
    }
    return array;
}

private static string[] Bucket(string[] array)
{
    List<string>[] buckets = new List<string>[26];

    foreach (string element in array)
    {
        int number = char.ToUpper(element[0]) - 'A';

        if (buckets[number] == null)
        {
            buckets[number] = new List<string>();
        }
        buckets[number].Add(element);
    }

    int index = 0;
    foreach (List<string> bucket in buckets)
    {
        if (bucket != null)
        {
            string[] insertion = Insertion(bucket.ToArray());

            foreach (string element in insertion)
            {
                array[index] = element;
                index++;
            }
        }
    }
    return array;
}

private static string[] Shell(string[] array)
{
    int gap = 1;

    while (gap < array.Length / 3)
    {
        gap = 3 * gap + 1;
    }

    while (gap > 0)
    {
        for (int i = gap; i < array.Length; i++)
        {
            string temporary = array[i];
            int index = i;

            while (index >= gap && string.Compare(array[index - gap], temporary)
> 0)
            {
                array[index] = array[index - gap];
                index -= gap;
            }

            array[index] = temporary;
        }

        gap /= 3;
    }

        return array;
}

private static int Binary(string[] array, string name)
```

```
{
    int start = 0;
    int end = array.Length - 1;

    while (start <= end)
    {
        int mid = (start + end) / 2;

        if (array[mid] == name)
        {
            return mid;
        }
        else if (String.Compare(name, array[mid]) < 0)
        {
            end = mid - 1;
        }
        else
        {
            start = mid + 1;
        }
    }
    return -1;
}

static void Main(string[] args)
{
    string[] array = File.ReadAllLines("names.txt");

    string[] bubble = Bubble(array);
    Console.WriteLine(Binary(bubble, "Wally"));

    string[] selection = Selection(array);
    Console.WriteLine(Binary(selection, "Wally"));

    string[] insertion = Insertion(array);
    Console.WriteLine(Binary(insertion, "Wally"));

    string[] bucket = Bucket(array);
    Console.WriteLine(Binary(bucket, "Wally"));

    string[] shell = Shell(array);
    Console.WriteLine(Binary(shell, "Wally"));

}
```

Bucket Sort: Sorting the names based on their first character using bucket sort.

- Create 26 buckets, one for each letter of the alphabet.
- The names are then distributed into the corresponding buckets based on the first character.
- Then, we sort each bucket separately using insertion sort.
- Then, we combine the buckets to get the final sorted list of names.

Shell sort:

- Compare elements that are a distance apart rather than adjacent.
- Calculate the gap for each pass.
- Then select the elements towards the right of gap.

End.