

```
1 using System;
2
3 namespace Unit4
4 {
5     class StackUtils
6     {
7         public static Stack<T> CreateStackFromArray<T>(T[] arr)
8         {
9             Stack<T> s = new Stack<T>();
10
11             return s;
12         }
13         public static Stack<T> CreateStackFromArrayEnd<T>(T[] arr)
14         {
15             Stack<T> s = new Stack<T>();
16
17             return s;
18         }
19
20         public static void SpilledOn<T>(Stack<T> to, Stack<T> from)
21         {
22         }
23         public static Stack<T> Clone<T>(Stack<T> s)
24         {
25             Stack<T> t = new Stack<T>();
26             Stack<T> s2 = new Stack<T>();
27             return s2;
28         }
29         public static Stack<T> CloneRec<T>(Stack<T> s)
30         {
31             return new Stack<T>();
32         }
33         public static Stack<T> ReverseStack<T>(Stack<T> s)
34         {
35             Stack<T> t = new Stack<T>();
36             Stack<T> t2 = new Stack<T>();
37             return t2;
38         }
39         public static int GetSize<T>(Stack<T> s)
40         {
41             return 0;
42         }
43         public static int GetSizeRec<T>(Stack<T> s)
44         {
45             return 0;
46         }
47         public static int Sum(Stack<int> s)
48         {
49             return 0;
```

```
50     }
51     public static int SumRec(Stack<int> s)
52     {
53         return 0;
54     }
55     public static bool IsSorted(Stack<int> s)
56     {
57         return false;
58     }
59     public static void Sort(Stack<int> s)
60     {
61     }
62
63     public static bool IsSortedRec(Stack<int> s)
64     {
65         return false;
66     }
67     public static bool IsExist(Stack<int> s, int val)
68     {
69         return false;
70     }
71     public static bool IsExistRec(Stack<int> s, int val)
72     {
73         return false;
74     }
75 }
76 }
77
```