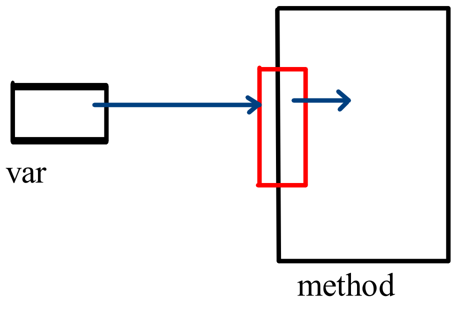
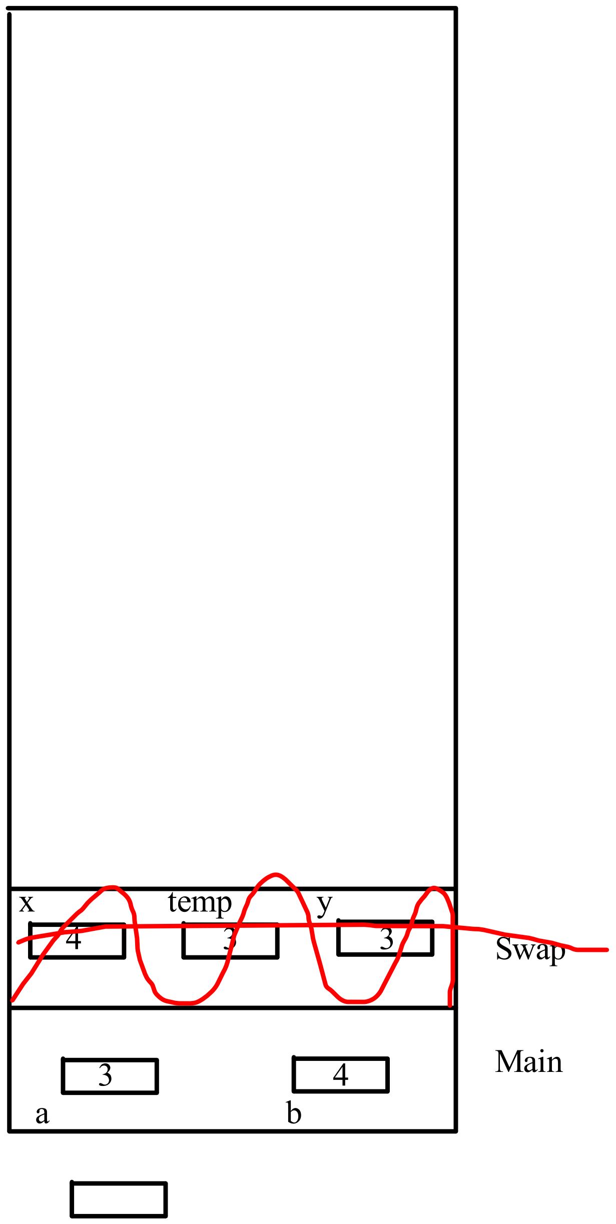
```
class MyClass
{
    public static void Main()
    {
        int a = 3, b = 4;
        Swap(a, b);
        Console.WriteLine($"A = {a}");
        Console.WriteLine($"B = {b}");
    }
    public static void Swap(int x, int y)
    {
        int temp;
        temp = x;
          x  =  y;
          y  = temp;
    }
}
```

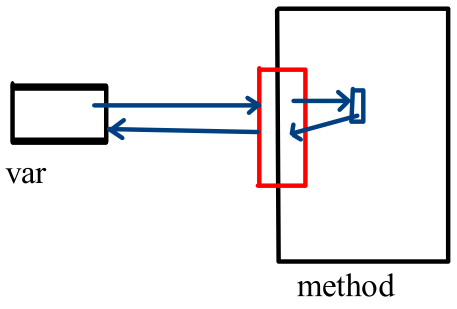
Call by Value
(in parameter)

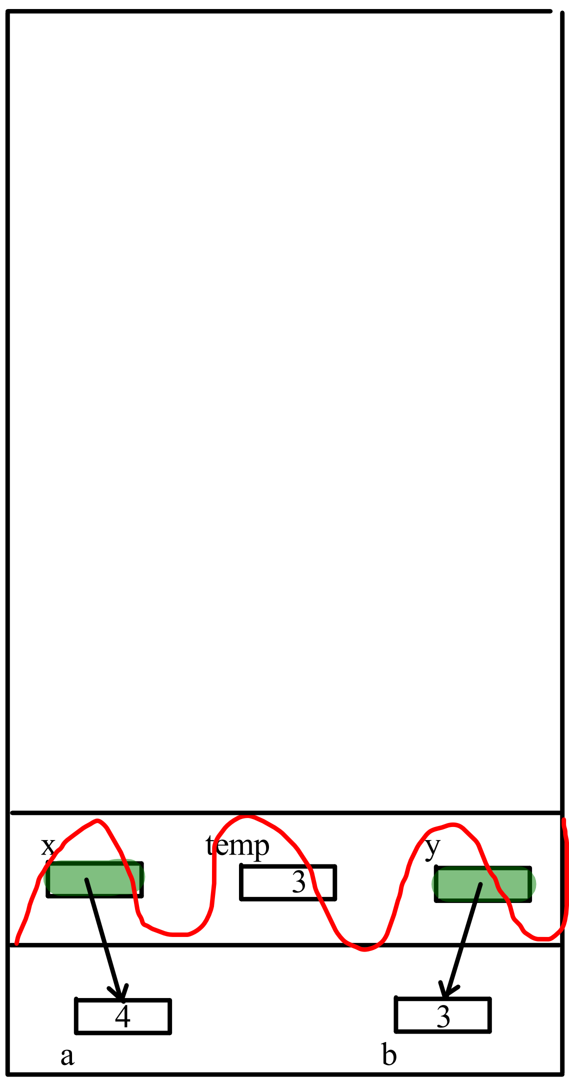




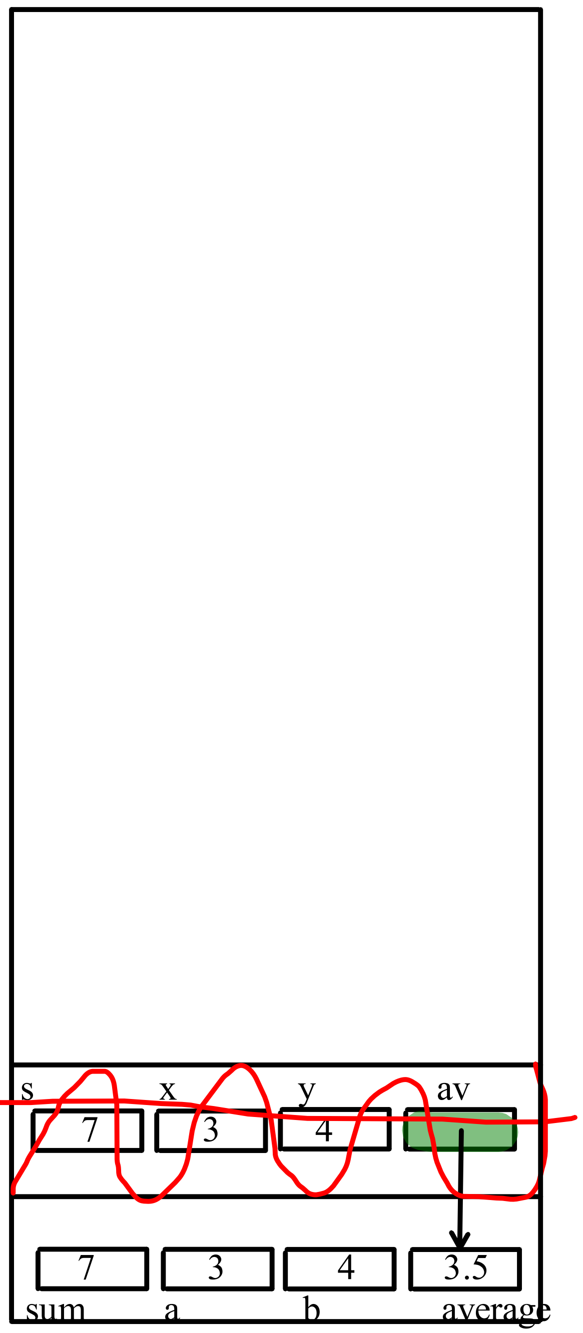
```
class MyClass
{
    public static void Main()
    {
        int a = 3, b = 4;
        Swap(ref a, ref b);
        Console.WriteLine($"A = {a}");
        Console.WriteLine($"B = {b}");
    }
    public static void Swap(ref int x, ref int y)
    {
        int temp;
        temp = x;
          x  =  y;
          y  = temp;
    }
}
```

Call by Reference
(in/out parameter)

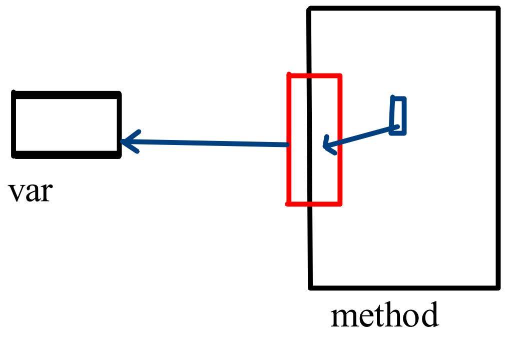




```
class MyClass
{
        public static void Main()
        {
                int a = 3, b = 4;
                float average;
                int sum;
                sum = Calc(a, b, out average);
                Console.WriteLine($"Sum = {sum}");
                Console.WriteLine($"Average = {average}");
        }
        public static int Calc(int x, int y, out float av)
        {
                int s;
                s = x + y;
                av = s/2.0;
                return s;
        }
}
```



out parameter:
is like ref parameter except:
out parameter act as uninitialized in the method (even it has prior value)
So, it must has a value in the method



```
class MyClass
{
        int[] ar = new int[5];  //assume +ve
        string name;

    public void SetName(string s)
    { name = s;}
    public string GetName()
    {return name;}

    public void SetArray(int index, int m)
    {
        if(index >= 0 && index < ar.Length)
        {
                ar[index] = m;
        }
    }
    public int GetArray(int index)
    {
        int val = -1;
        if(index >= 0 && index < ar.Length)
        {
                val = ar[index];
        }
        return val;
    }

}

class test
{
   public static void Main()
   {
        MyClass obj = new MyClass();
        obj.SetArray(3, 7);
        obj[3] = 7;
   }
}
```

```
class MyClass
{
        int[] ar = new int[5];  //assume +ve
        string name;

    public void SetName(string s)
    { name = s;}
    public string GetName()
    {return name;}
//Indexer
    public int this[int index]
    {
        set
        {
                if(index >= 0 && index < ar.Length)
                {
                        ar[index] = value;
                }
        }
        get
        {
                int val = -1;
                if(index >= 0 && index < ar.Length)
                {
                        val = ar[index];
                }
                return val;
        }
    }

}
```