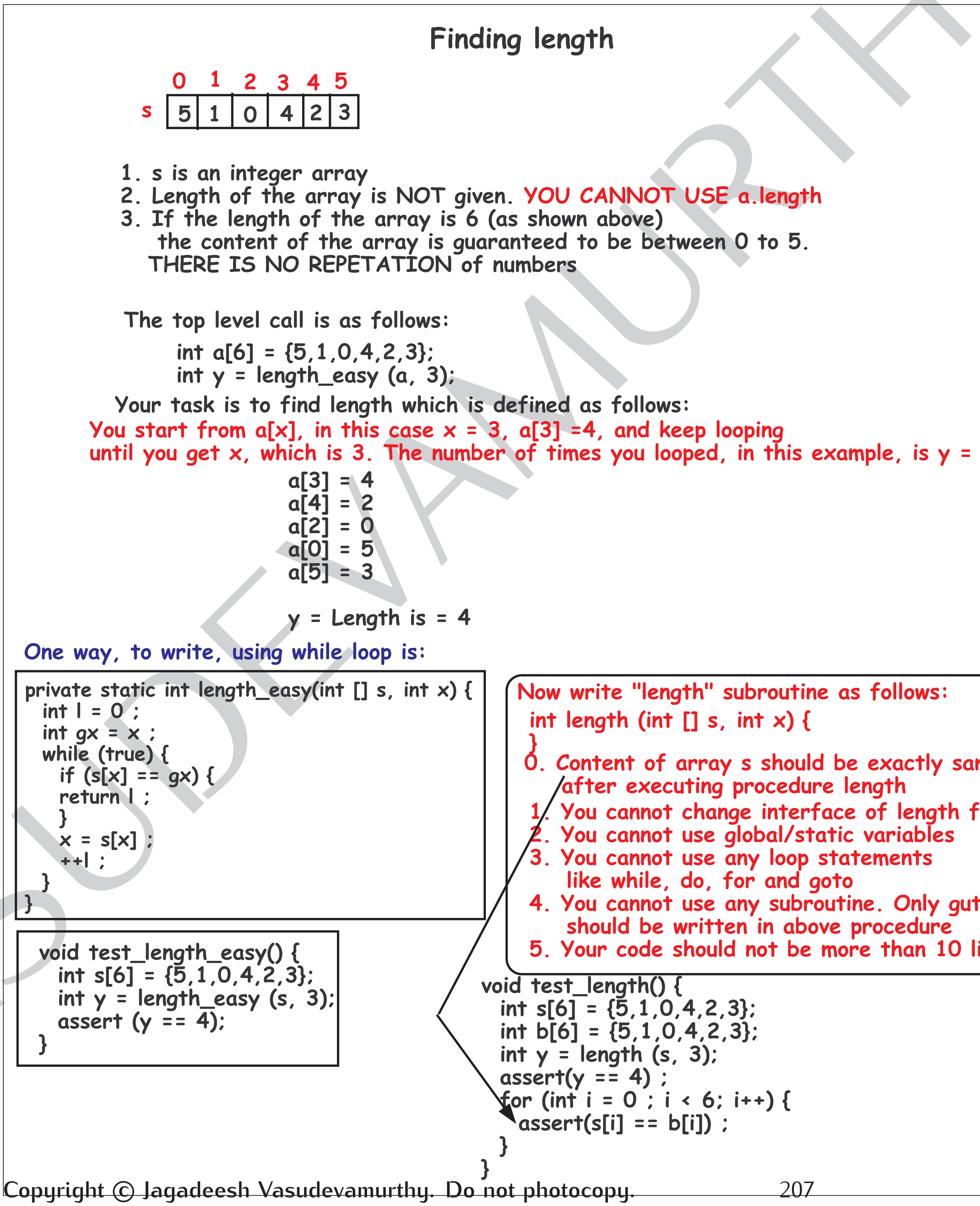
3.2 Problem set

Problem 3.2.1. Write a procedure called **length** as shown in the figure 3.1. Show on paper, how your code works. Note that you cannot use any loop statements and array length is not known.



Finding length

	0	1	2	3	4	5
s	5	1	0	4	2	3

1. s is an integer array
2. Length of the array is NOT given. YOU CANNOT USE a.length
3. If the length of the array is 6 (as shown above)
   the content of the array is guaranteed to be between 0 to 5.
   THERE IS NO REPETATION of numbers

The top level call is as follows:

```
int a[6] = {5,1,0,4,2,3};
int y = length_easy (a, 3);
```

Your task is to find length which is defined as follows:
You start from a[x], in this case x = 3, a[3] =4, and keep looping
until you get x, which is 3. The number of times you looped, in this example, is y =

```
a[3] = 4
a[4] = 2
a[2] = 0
a[0] = 5
a[5] = 3

y = Length is = 4
```

One way, to write, using while loop is:

```
private static int length_easy(int [] s, int x) {
  int l = 0 ;
  int gx = x ;
  while (true) {
    if (s[x] == gx) {
    return l ;
    }
    x = s[x] ;
    ++l ;
  }
}
```

```
  void test_length_easy() {
    int s[6] = {5,1,0,4,2,3};
    int y = length_easy (s, 3);
    assert (y == 4);
  }
```

Now write "length" subroutine as follows:

```
int length (int [] s, int x) {
}
```

0. Content of array s should be exactly sa
   after executing procedure length
1. You cannot change interface of length f
2. You cannot use global/static variables
3. You cannot use any loop statements
   like while, do, for and goto
4. You cannot use any subroutine. Only gut
   should be written in above procedure
5. Your code should not be more than 10 li

```
void test_length() {
  int s[6] = {5,1,0,4,2,3};
  int b[6] = {5,1,0,4,2,3};
  int y = length (s, 3);
  assert(y == 4) ;
  for (int i = 0 ; i < 6; i++) {
   assert(s[i] == b[i]) ;
  }
}
```

Figure 3.1: Finding the length