(a)

Data.hasNext()	A	A >=0	A mod 2 = 0	Contents of B	Contents of C
true	2	true	true (2 mod 2 =0)	2	
true	4	true	true(4 mod 2 = 0)	2, 4	
true	-1	not true	not true (-1 mod 2 = 1)	2,4	
true	3	true	not true	2,4	3

(b)

- Determine a search value
- Calculate the mid value of the array between the high and low values
- If the mid value of the array is equal to the search value, then the search value is determined and the loop can be ended
- If the search value is greater than the mid value in the array, a new mid value must be set at the lower end of the array. Similarly, the mid value of the lower end of the array will be compared with the search element. If both values are the same, then the user can exit the loop. However, if it is not found then the user can now find the mid value in the higher end of the array. Same process will repeat.
- If the user finds that the mid value matches the search element, then the user can exit the program.

(c)

Algorithm

```
Input S
Count = 0
FOUND = false
Start while loop NUMBERS.hasNext ( )
        D [COUNT] = NUMBERS.getNext ( )
        If S ==D [COUNT]
end if
COUNT = COUNT + 1

end loop
if FOUND

output “found”
```

else

output “not found”

end if

(d)

Without writing a pseudo code, this check can be performed simply by comparing the current value in the array with the previous one. Additionally, to check whether the array is sorted or not, we can start from the second value and compare it with the values that are being entered. Lastly, we can also check the order by setting and changing the flag variable. For instance, if the array variable is less than the search variable then it is not in ascending order. At that time, if we put flag variable as 1 then it can be determined that it is not in ascending order and that we must keep changing it until the array is sorted.