Computational Thinking with Algorithms Problem Sheet (Java)

Question 2 (8 marks)

Consider the following methods:

```
public static int finder(int[] input) {
    return finderRec(input,input.length-1);
}

public static int finderRec(int[] input, int x) {
    if(x==0) {
        return input[x];
    }

    int v1 = input[x];
    int v2 = finderRec(input, x-1);

    if(v1>v2) {
        return v1;
    }
    else {
        return v2;
    }
}
```

Q2 (a) What is the output of a call to `finder` when the following array is used as input? **(1 mark)**

[0,-247,341,1001,741,22]

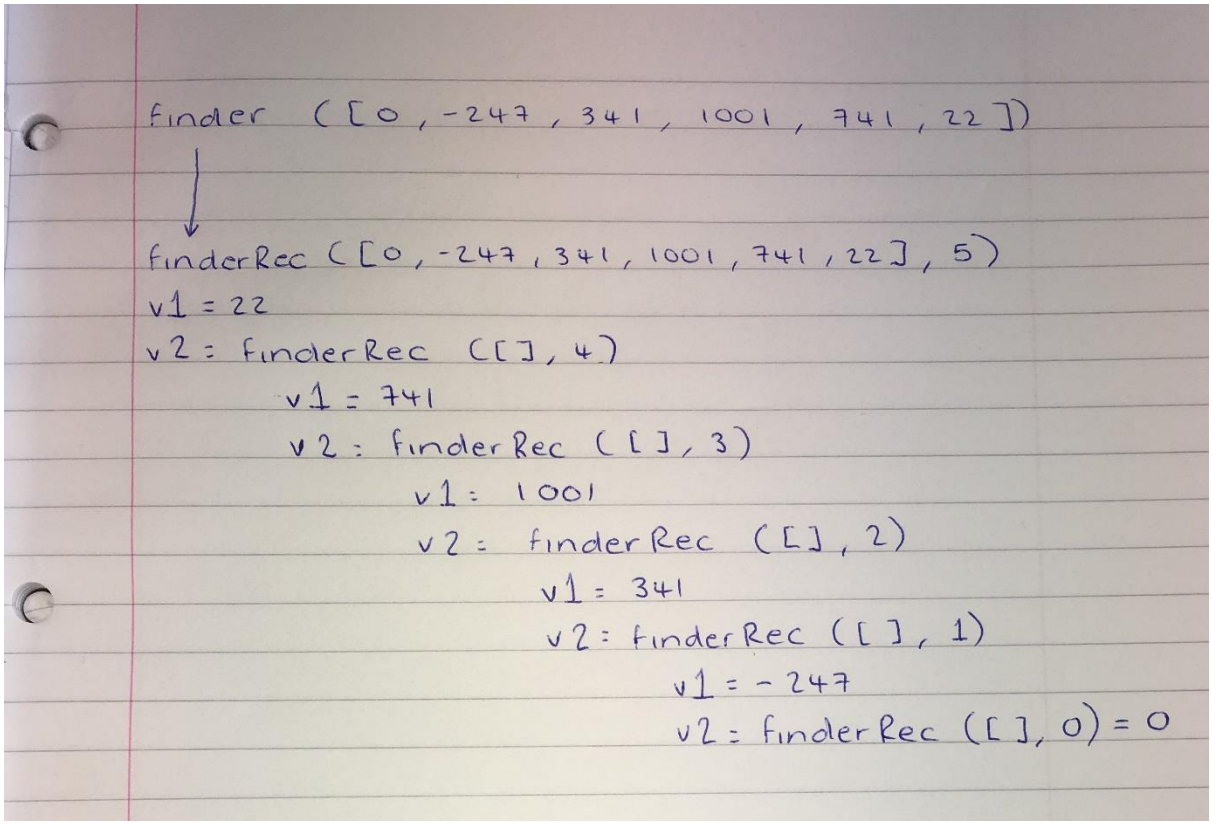
Q2 (b) What characteristic of the input data set does the `finder` method determine? How does it determine this result? **(3 marks)**

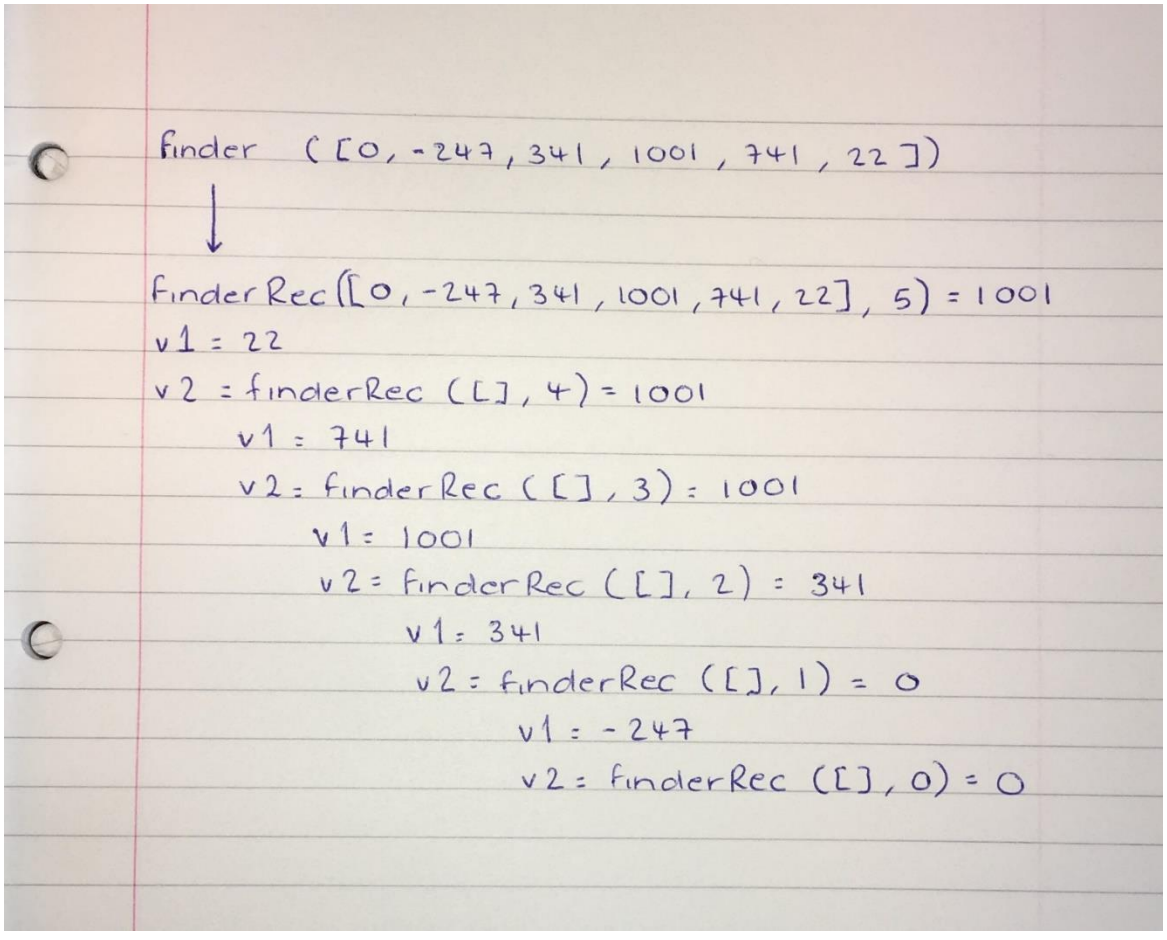
Q2 (c) Can you add some inline comments to the code above to explain how it works? **(2 marks)**

Q2 (d) Write a method which achieves the same result as `finder`, but which uses an iterative approach instead of recursion. **(2 marks)**

Write an explanation of the reasoning behind your answers to the above questions, using the aid of either a recursion trace diagram or a stack diagram for Q2 (b). Include any code which you write for testing or explanation purposes as part of your answer.

Recursion Trace Diagrams for Question 2:





Q2 (a)

Output: 1001

Q2 (b)

The finder method is determining the maximum value stored in the array, recursively.

See in the recursion trace above if we work our way from the bottom up –

1. Initially we compare element 0 (value=0) with element 1 (value= -247). We return the largest value i.e. value=0.
2. Secondly we take the element 0 (value=0) and compare it with element 2 (value= 341). The maximum is 341, so we return that value.
3. Thirdly we compare element 2(value=341) with element 3 (value=1001). The maximum value is 1001, so we return that value.
4. Next we compare element 3 (value=1001) with element 4(value= 741). The maximum value is 1001, so we return that value.
5. Fiinally we compare element 3(value=1001) with element 5(value=22). The maximum value is 1001, so we return that value.

Since we have made the comparison for each of the elements, the result is the maximum value of the array, which is element 3 (value=1001).

Q2 (c) Inline Comments:

finder() function:

Line 1: This function converts the input into a version compatible with the recursive fn.

Line 3: It does this by passing the length – 1 arguments into the recursive fn. Length-1 points to the last element in the array.

finderRec() function:

Line 2: This is the termination condition/base case. We reach this condition when making the last statement.

Line 3: Return the 0^{th} element.

Line 5: (int v1) The next element of the array to compare.

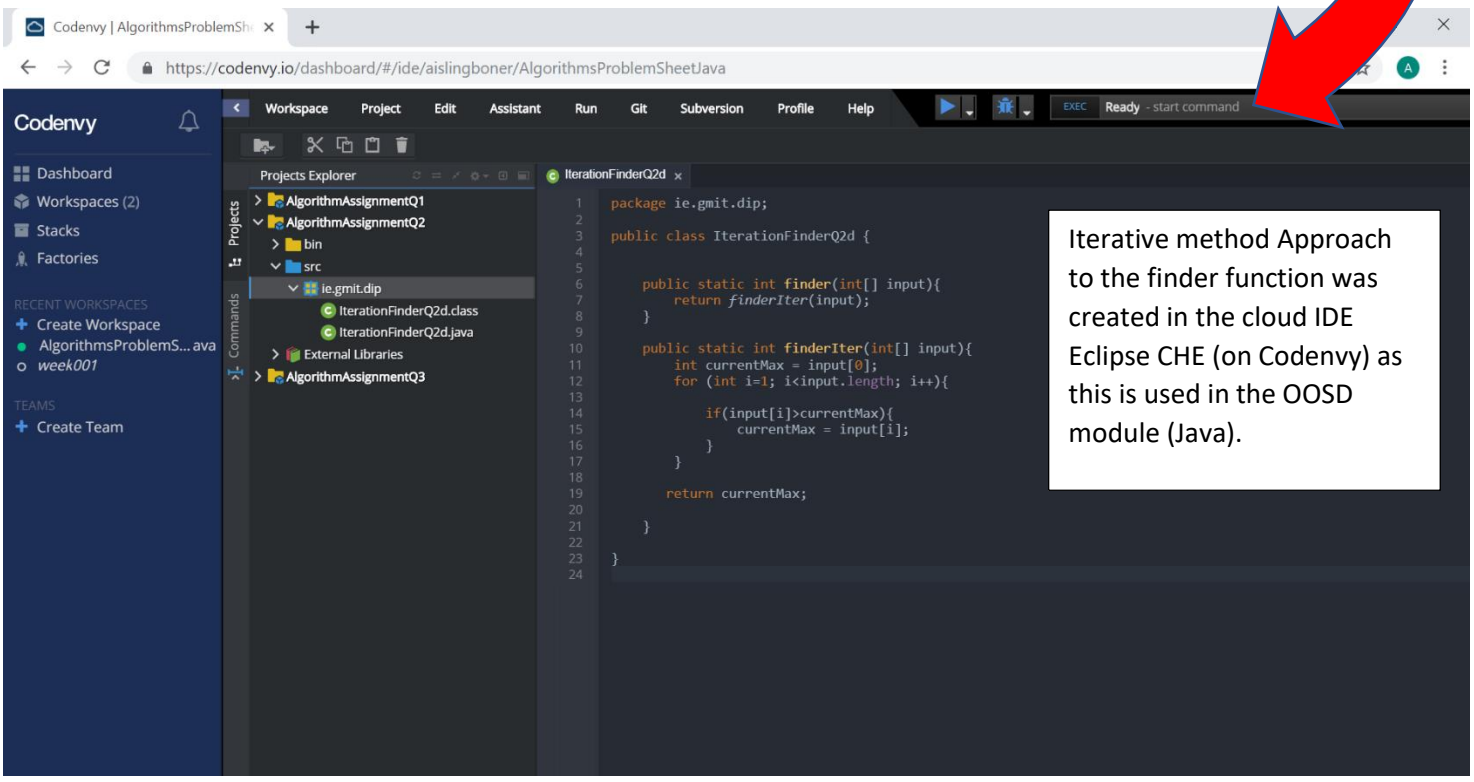
Line 6: Recursive call, which returns the max value of the subset of the array, from elements 0 to x.

Line 8: (if (v1>v2)) Return the max value of v1 and v2.

Q2 (d) Iterative Method Approach instead of Recursive.

```
5
6       public static int finder(int[] input){
7           return finderIter(input);
8       }
9
10      public static int finderIter(int[] input){
11          int currentMax = input[0];
12          for (int i=1; i<input.length; i++){
13
14              if(input[i]>currentMax){
15                  currentMax = input[i];
16              }
17          }
18
19          return currentMax;
20
21      }
22
23  }
24
```

Iterative approach for the method finder which achieves the same result as the recursive approach.



Iterative method Approach to the finder function was created in the cloud IDE Eclipse CHE (on Codenvy) as this is used in the OOSD module (Java).