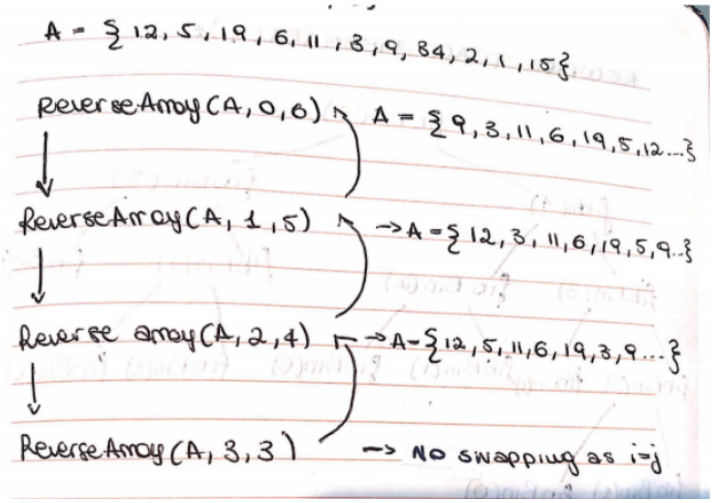
Recursion_WrittenQuestions

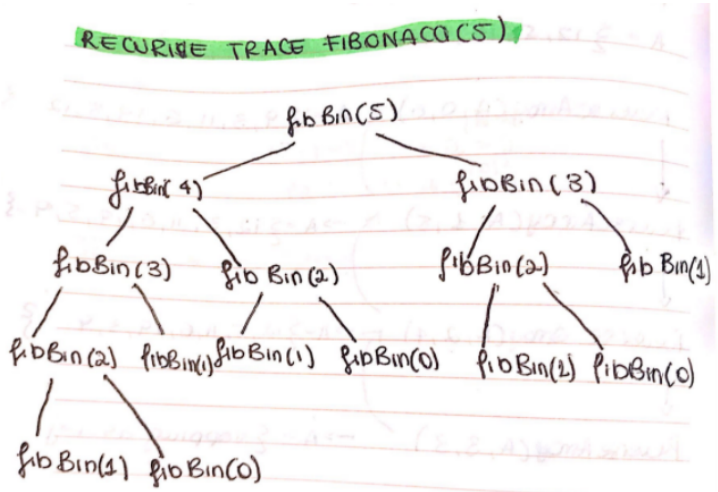
Question 1)

Draw the recursion trace for *ReverseArray(A, 0, 6)* where A = {12, 4, 19, 6, 11, 3, 9, 34, 2, 1, 15}



Question 2)

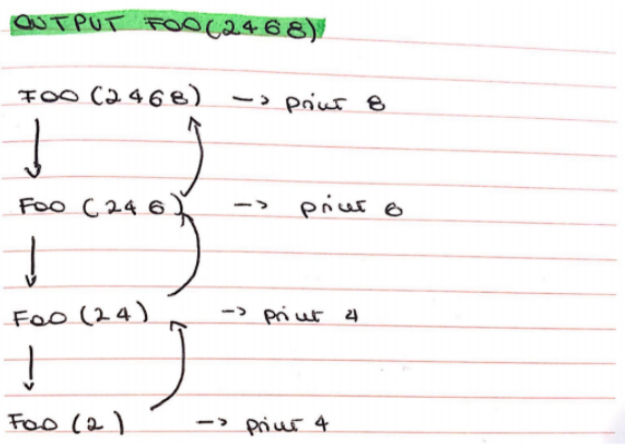
Using the binary recursive version of Fibonacci, write the recursive trace for Fibonacci(5)



Question 3)

a) The function *foo(int x)* recursively divides a number by 10, while the result is still a positive integer. It prints the remainder of the division by 10.

b) The output of calling *foo(2468)* would be the following:



Question 4)
Write a recursive function *sumOfDigits(int x)* that returns the sum of the digits in an integer, x.

```
METHOD sumOfDigits(int number)
      Input: the number whose digits the function needs to sum
      Output: the sum of the digits in the number.

      //initialize the variable sum as 0
      sum = 0

      //if there are any digits left in the number, add the remainder given
      //when dividing the number by 10 and make the recursive call
      if (number/10) != 0, then
            sum ← sum + number mod 10
            sumOfDigits(number/10)

return sum
```

Question 5)
a) Write a recursive function which prints the elements of a linked list in reverse.

```
METHOD printReverse(Node head)
      Input: the head node of the SinglyLinkedList

      //base case: if the head node is null, there is nothing to do
      if (head == null), then
            return

      //recursive call: if the head node is not null,
      //traverse the list in a recursive manner and print each element
      //note: due to the nature of recursion, the list will be printed in
      //reverse order
      printReverse(head.next)
      print head.element

return
```

Question 6)
a) Write a recursive function which copies all the elements in a linked list

```
METHOD copyRecursive(Node head)
      Input: the head node of the SinglyLinkedList
      Return: the new linked list (a copy of the first)

      new LinkedList copy
```

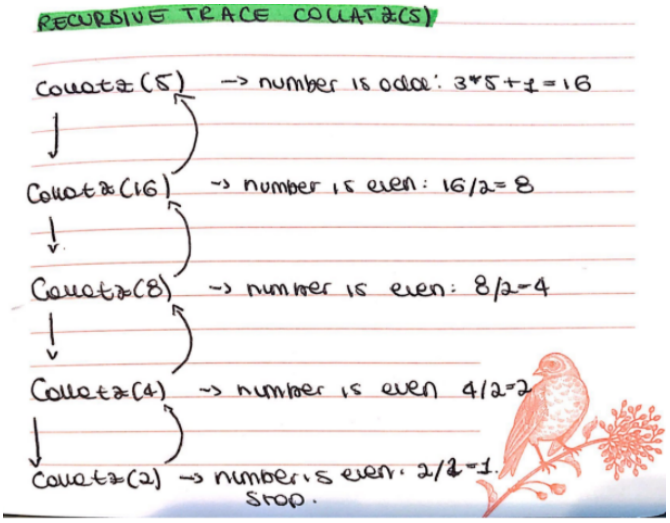
```
        //base case: if the head node is null, return the list
        if (head == null), then
              return copy

        //recursive call: if the head node is not null,
        //traverse the list in a recursive manner and add each element to the
        //new list
        copy.add ← head
        copyRecursive(head.next)

return copy
```

Question 6)

a) Write a recursive trace for Collatz(5)



b) The result of calling Collatz(9) is 1.

c) The result of calling Collatz(871) is 1.

Question 9)

Empirical Analysis - difference between bubble sort recursive and bubble sort iterative

Input Size (n)	Time elapsed - Bubble Sort Iterative	Time elapsed - Bubble Sort Recursive
1000	0.006	0.011
1500	0.002	0.005
2250	0.005	0.012
3375	0.013	0.013
5062	0.028	0.032

7593	0.071	0.088
11389	0.184	0.209
17083	0.398	0.496
25624	0.97	1.156
38436	2.173	2.748
57654	4.709	5.201
86481	10.748	12.823
129721	24.59	26.321
194581	56.768	57.322

Timing Graph: