1. Determine the output of the call recur(2).

```
void recur(int x) {
  if (x < 0) return;
  print(x);
  recur(x-1);
  recur(x-1);
}
```

2. Describe in plain words how the return value of function ggg() depends on the values of x and y.

```
int ggg(int x, int y) {
 if (x < y) return y;
 return ggg(y,x);
}
```

3 Describe in plain words how the return value of function ff depends on the values of x and y.

```
int ff(int x, int y) {
 if (x < y) return ff(x+1,y);
 return x;
}
```

4. Determine the result of the command **`print(recur(8));`**:

```
int recur(int x) {
  if (x < 1) return 2;
  return (recur(x-3)+recur(x-4));
}
```

5. Determine how long will it take to complete one call of **`recur(7);`** provided that a call **`xyz();`** takes always one millisecond and the time of other actions is neglected.

```
  void recur(int x) {
    if (x < 1) return;
    recur(x-3);
    xyz();
    recur(x-2);
  }
```

6. Write a recursive function which input is a positive integer N and the output is a sequence of integers:
1 2 ... N-2 N-1 N N N-1 N-2 ... 2 1.

7. Write a recursive function which input is a positive integer N and the output is a sequence of N 1's followed by 2N 2's. For example, when N = 3 the output would be 111222222.

8. The sequence 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 might be generated by a recrsive function called with parameter 4:

```
  void ruler(int val) {
    if (val < 1) return;
    ruler(val-1);
    printf("%d%s",val," ");
    ruler(val-1);
  }
```

How many integers and how many characters contains the output when the parameter is 20?

9. Write a recursive function which returns the index of the rightmost maximum value of the input 1D array. The array will be the parameter of the function. Decide which other parametrs would be necessary.

10. Write a recursive function which returns the number of non-zero values in the input 1D array. The function should call itself twice in its body. The complexity of the function should be linear with respect to the size of the input array.

11. Recursive algorithm A divides the problem of size n into 2 parts of the same size, to get a result each of the parts must be processed twice. The time required to divide the problem into two parts and to merge the partial solutions is proportional to n. Asymptotic complexity of A is described by the formula:

a) $T(n) = 4T(n/2) + n$
b) $T(n) = n \cdot T(n \cdot 4/2)$
c) $T(n) = T(n/2) + 4n/2$
d) $T(n) = 2T(n/4) + n$
e) $T(n) = n \cdot T(n/2) + n \cdot \log(n)$

12. Recursive algorithm A divides the problem of size n into 4 parts of the same size, to get a result only three of the parts must be processed (each is processed once). The time required to divide the problem into parts and to merge the partial solutions is proportional to n. Asymptotic complexity of A is described by the formula:

$T(n) = 4T(n/3) + n$
$T(n) = n \cdot T(n \cdot 4/3)$
$T(n) = 4T(3n) - n$
$T(n) = 3T(n/4) + n$
$T(n) = n \cdot T(n/3) + n \cdot \log(n)$

13. Recursive algorithm A divides the problem of size n into 4 parts of the same size, to get a result three of the parts must be processed once and the fourth part must be processed twice. The time required to divide the problem into parts and to merge the partial solutions is proportional to $n^2 - n$.

a. Draw first three levels (the root and two levels bellow) of the recursion tree.

b. Suppose that the root corresponds to processing some input data of size n.
Find the cost of the nodes in the depth 2 (3rd level including the root). The cost of a node is equal to the time which the algoruithm spends on dividing the problem into subproblems and merging the partial solutions. The size of the data associated wirh a particular node corresponds to the depth of the node.

c. Find the depth of the recursion tree.

d. Use Master theorem to determine the complexity of the algorithm.

14. Solve the previous problem with different parameters, complete the steps a., b., c., d. analogously.
The algorithm divides the data into 3 parts of the same size, each part processes twice and then merges the results.
The time required to divide the problem into parts and to merge the partial solutions is proportional to $\sqrt{n} \cdot \log_2(n)$.